Introduction to Machine Learning

- Main concepts in artificial intelligence and machine learning
- Types of learning and available libraries
- Hands-on exercises on modelization and optimization
- Data competition: challenge your knowledge!
- (Optional) Bring your own problem and get help onsite

Programme

09:00-10:30: Introduction ML
10:30-11:45: break
10:45-12:00: Practical session
12:00-13:00: Lunch break
13:00-14:30: data modeling
14:30-14:45: break
14:45-17:00: data challenge

About SURFsara

History:

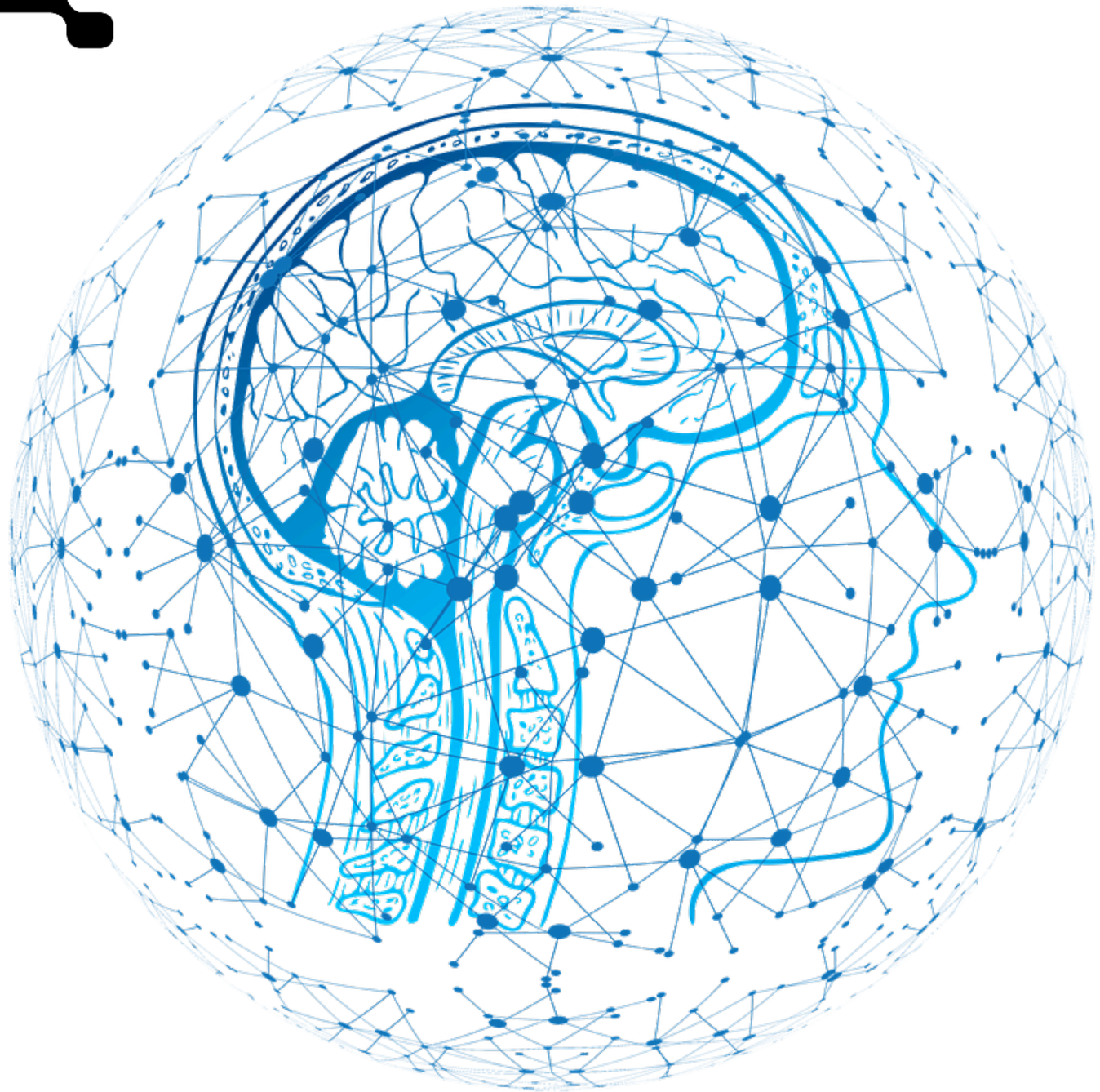
- 1971: Founded by the VU, UvA, and CWI
- 2013: SARA (Stichting Academisch Rekencentrum A'dam) becomes part of SURF

Super/cluster-computing group:

- 8 consultants
- 16 members in total (including admins/system-experts)

Other activities:

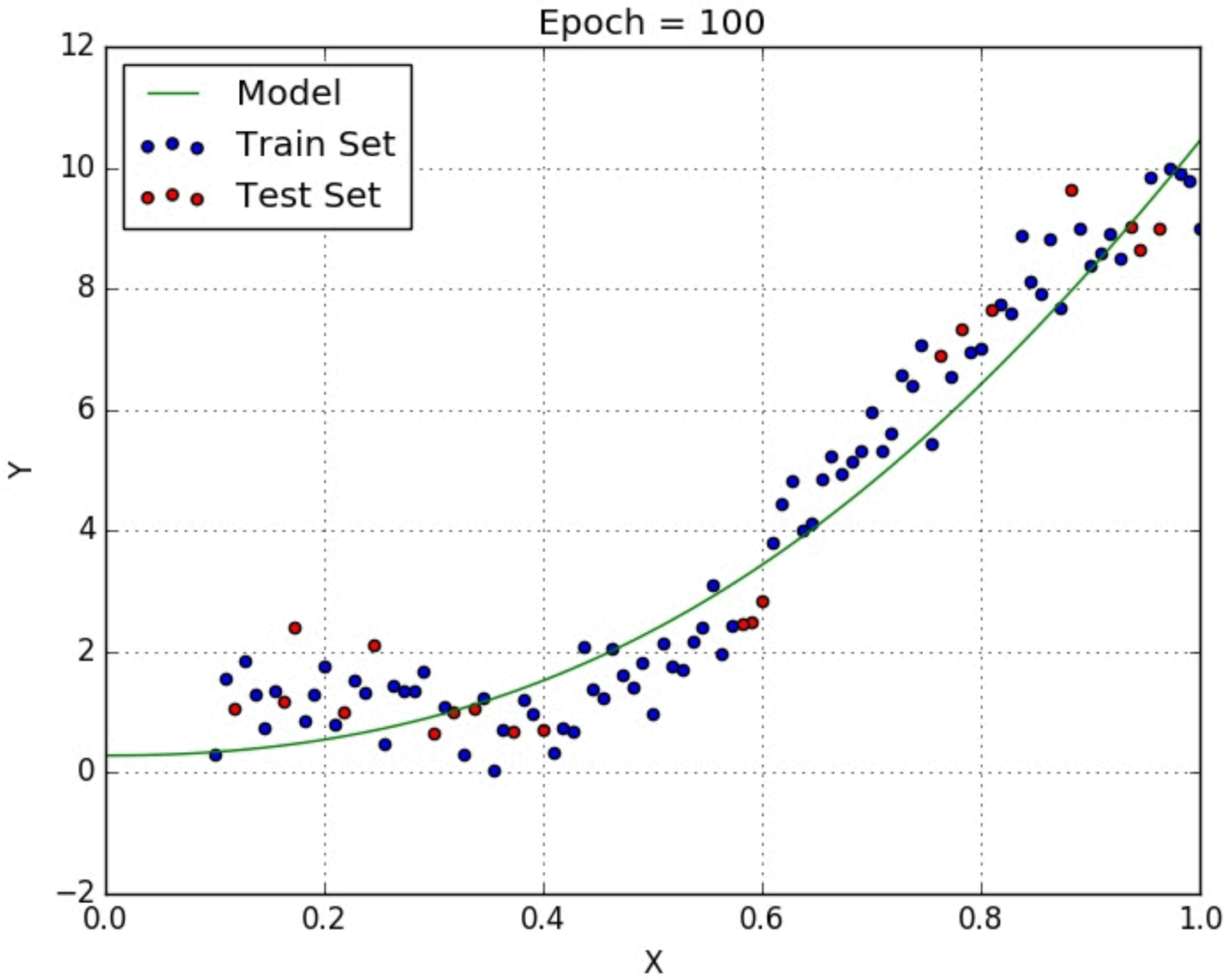
- HPC cloud / virtualisation
- Big Data
- Data services / storage
- Visualisation



Introduction to Machine Learning & Deep Learning

Maxwell Cai (SURFsara)

What *is* machine learning?

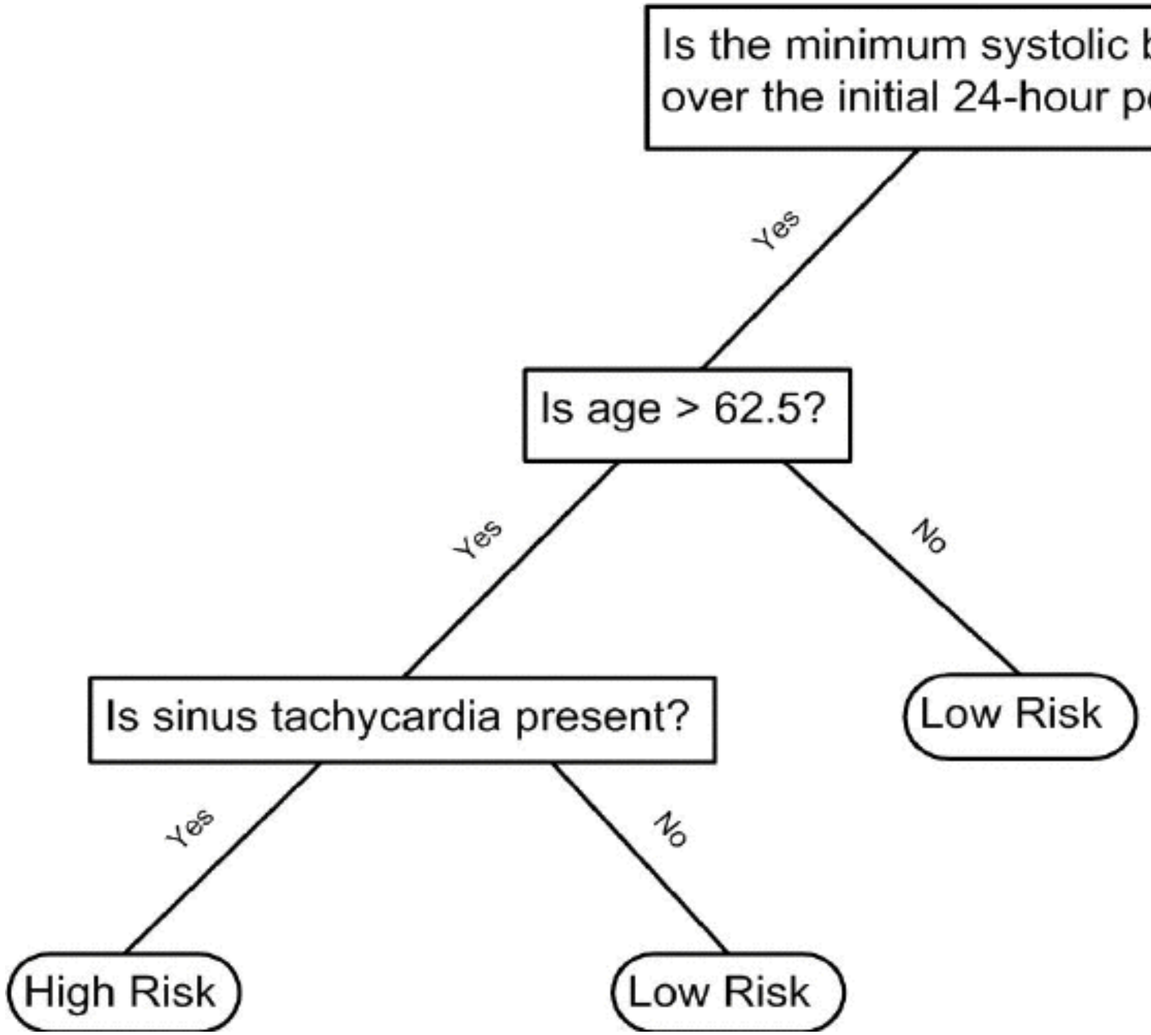


Could be as simple as curve fitting!

Typical applications in general:

- Classification
- Regression
- Dimensionality reduction
- Control

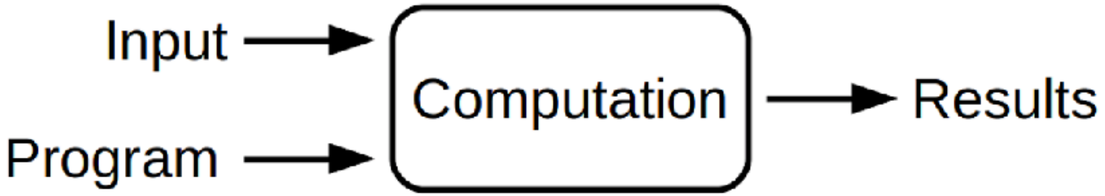
Decision Tree(s)




```
if blood_pressure > 91:
    …
else:
    …
```

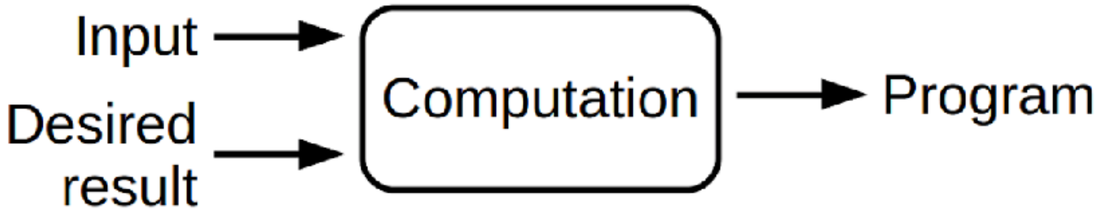
What *is* machine learning?

Traditional programming



Knowledge base & rules
Expert systems
Human input

Machine learning



Features and result
'Decision' system
Limited human input

Source: Machine Learning for an Expert System to Predict Preterm Birth Risk , Woolery et al. (1994)

AI vs ML vs DL

Artificial Intelligence

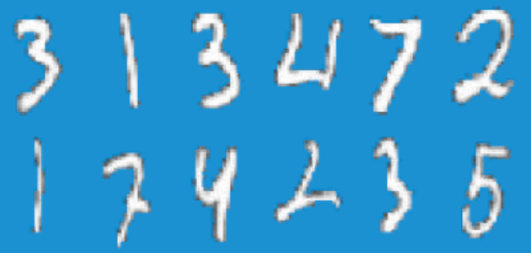
Any technique that enables computers to mimic human behavior

Machine Learning

Ability to learn without explicitly being programmed

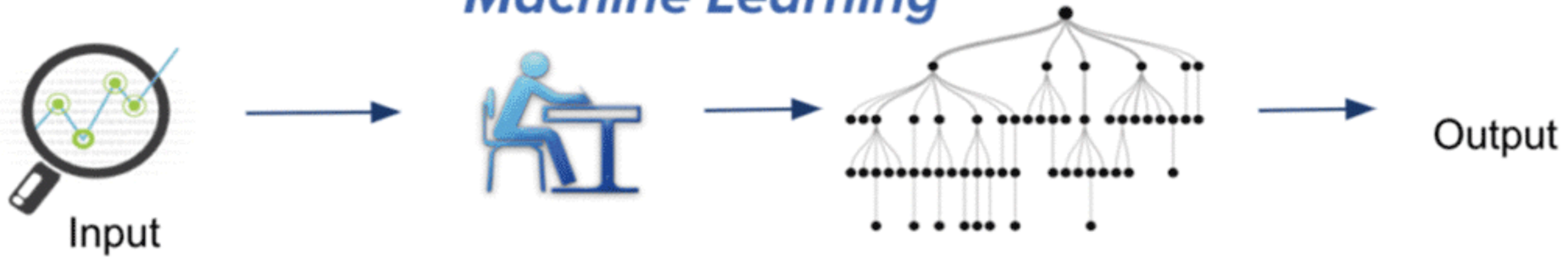
Deep Learning

Extract patterns from data using neural networks



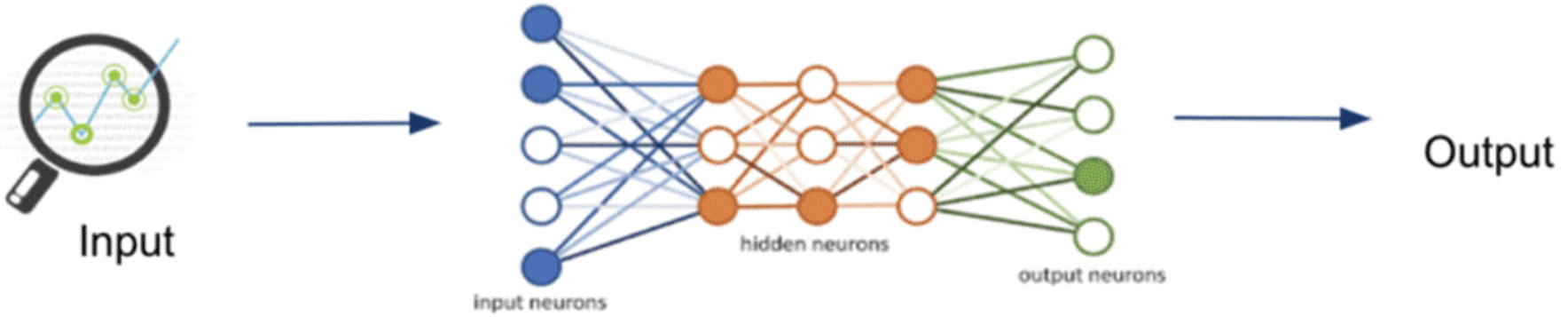
Credit: Alexander Amini/MIT

Machine Learning



Traditional machine learning uses hand-crafted features, which is tedious and costly to develop.

Deep Learning



Neural Networks

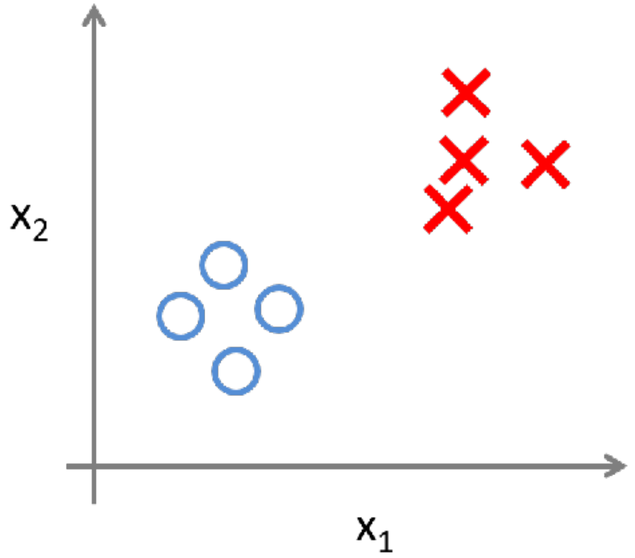
Deep learning learns hierarchical representation from the data itself, and scales with more data.

Source: idm.net.au

Categories of machine learning

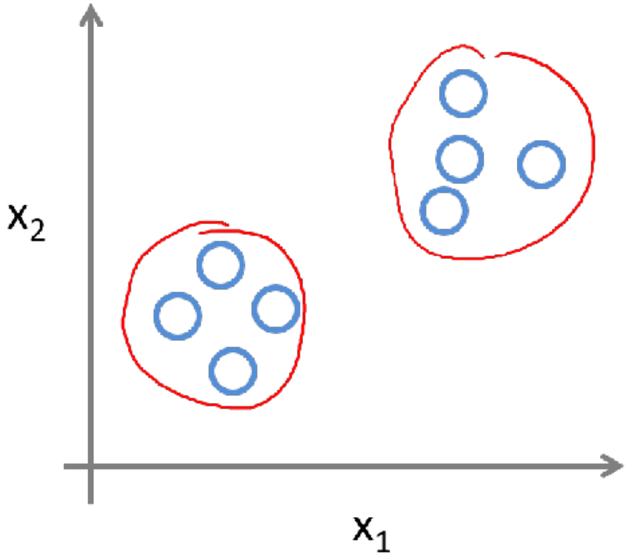
Supervised

Learn from the labels



Unsupervised

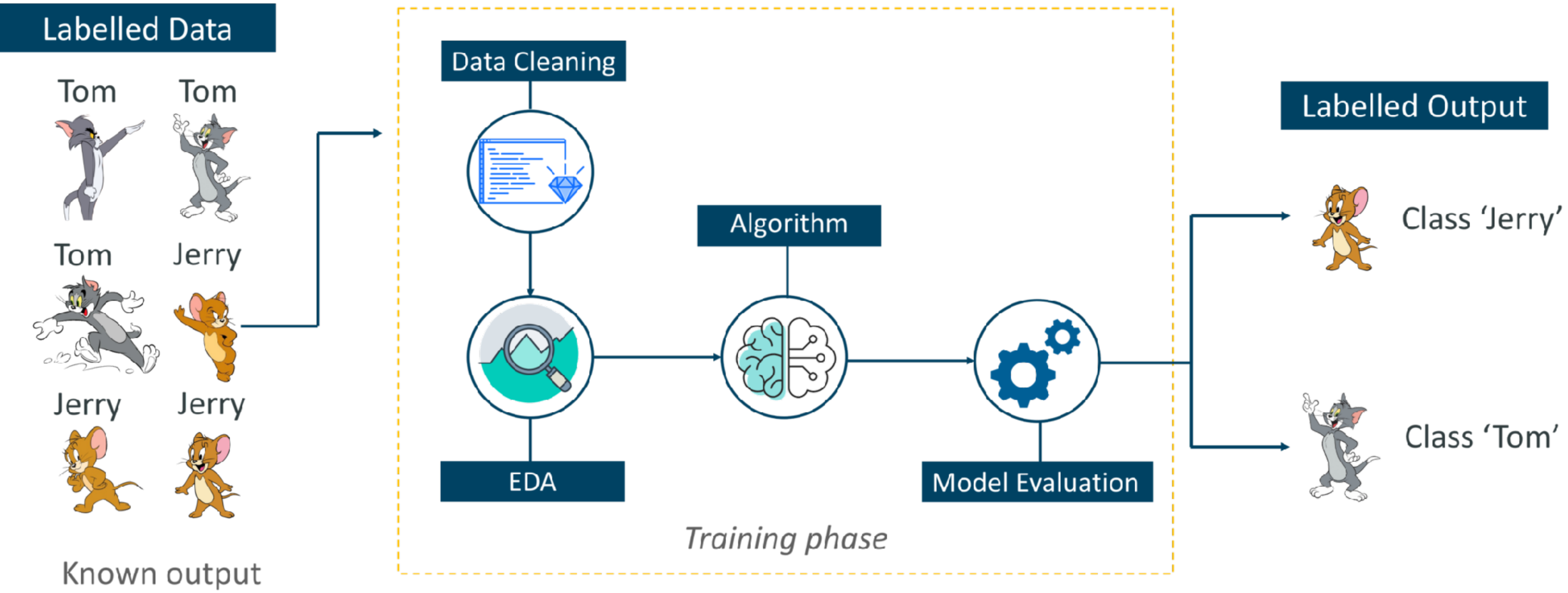
Detect patterns in the data



Reinforcement

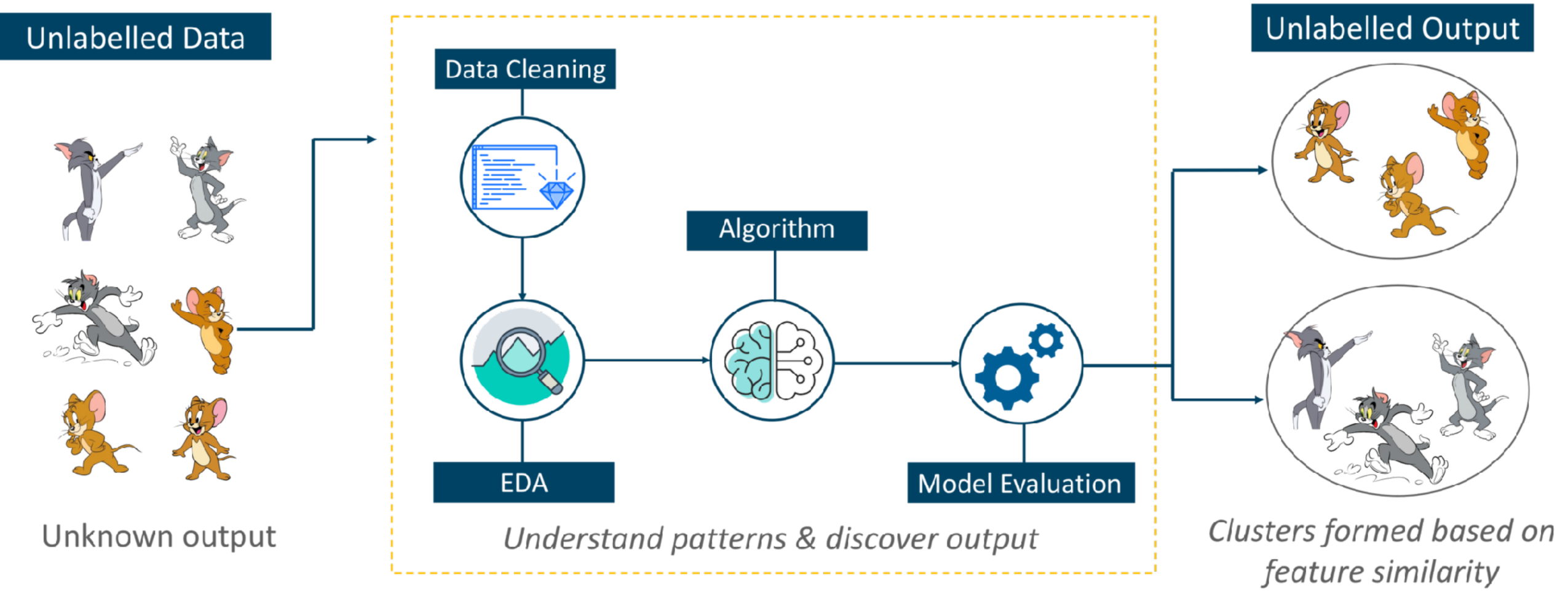
Learn from mistakes

Supervised Learning



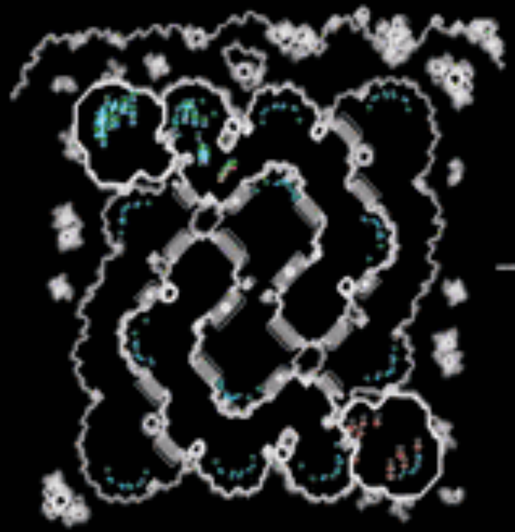
Source: edureka.co

Unsupervised Learning

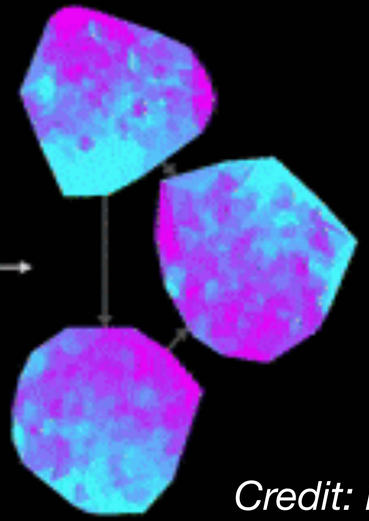


Source: edureka.co

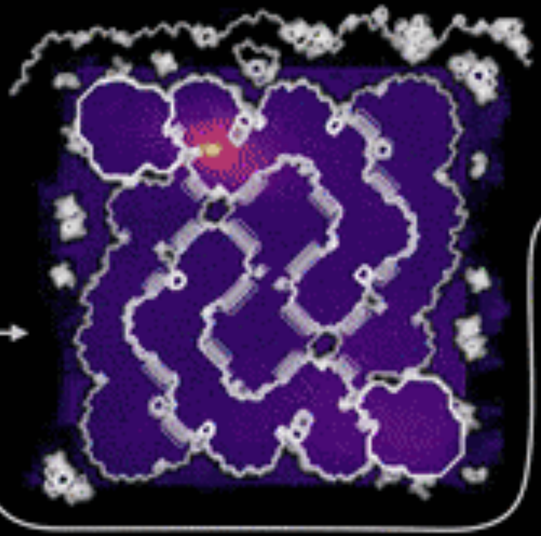
Raw Observations



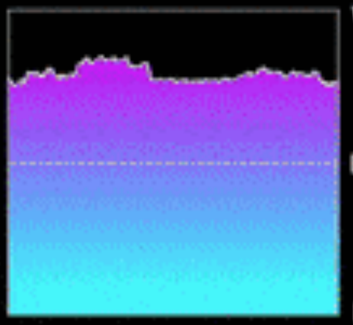
Neural Network Activations



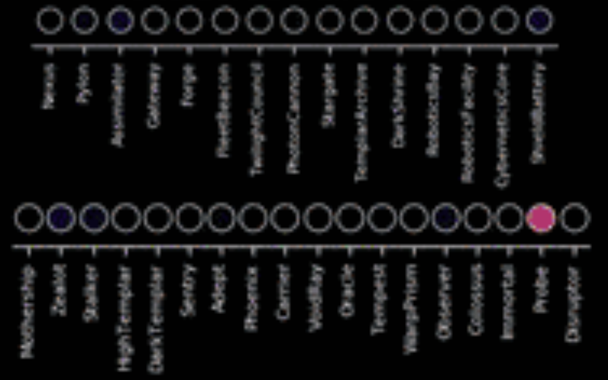
Considered Location



Outcome Prediction

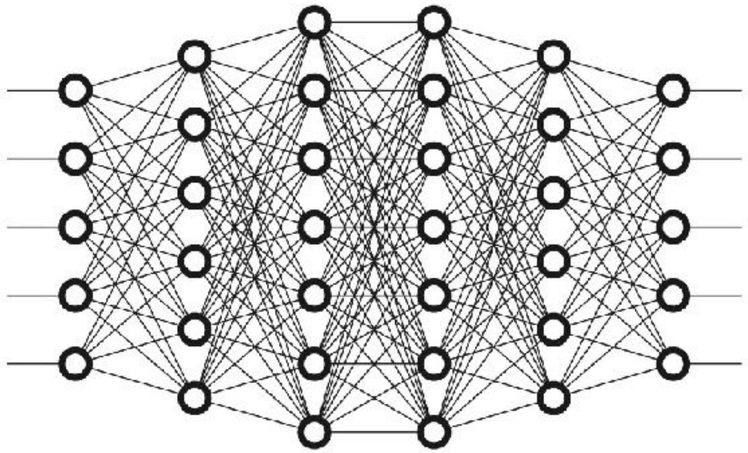


Considered Build/Train



Credit: Deepmind

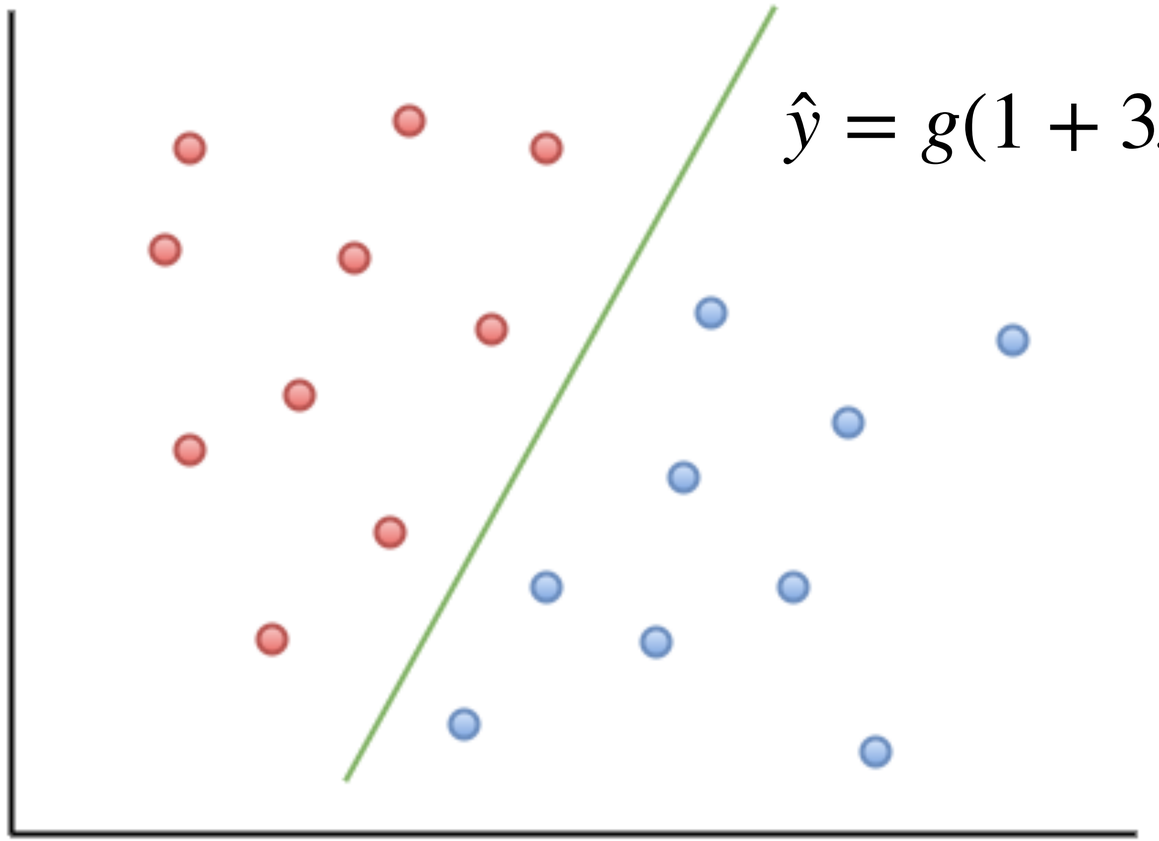
Input



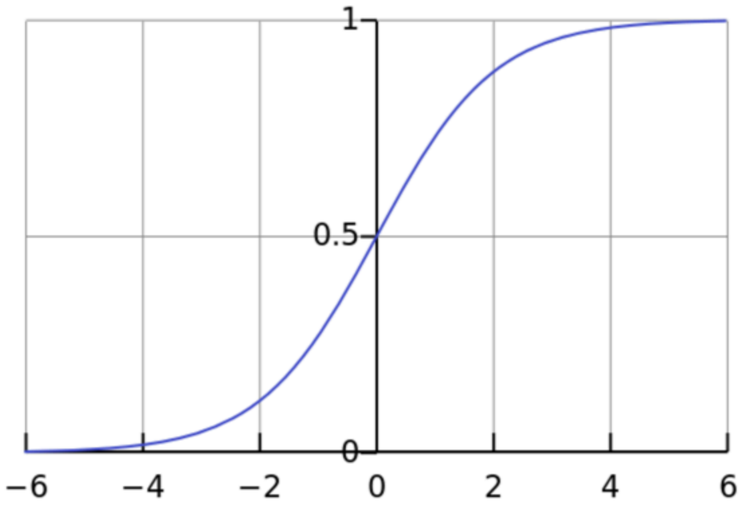
Output

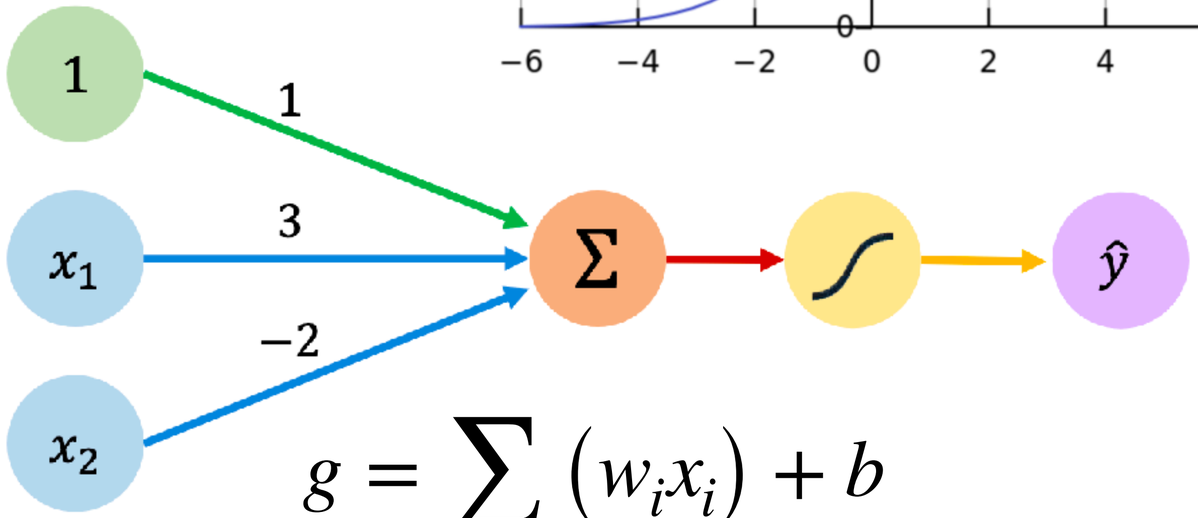
$$\hat{y} = f_{\mathrm{NN}}(x_1, x_2, \ldots, x_n)$$

Neurons/Perceptrons & Activation Functions



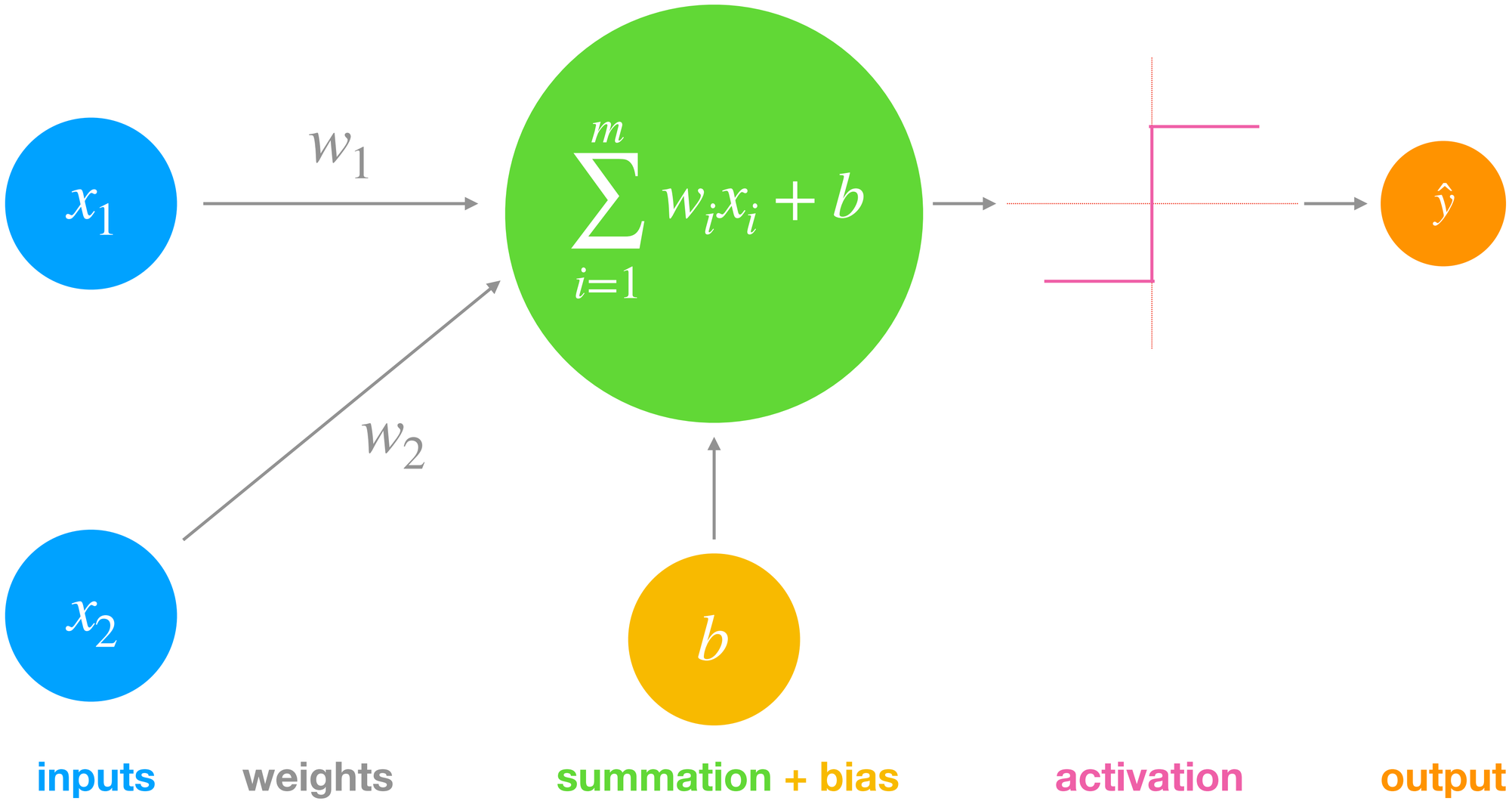
$$\hat{y} = g(1 + 3x_1 - 2x_2)$$





$$g = \sum (w_i x_i) + b$$

Prediction


x_1
w_1
$\sum_{i=1}^{m} w_i x_i + b$
$\hat{y}$
w_2
x_2
b
inputs
weights
summation + bias
activation
output

A few more terms…

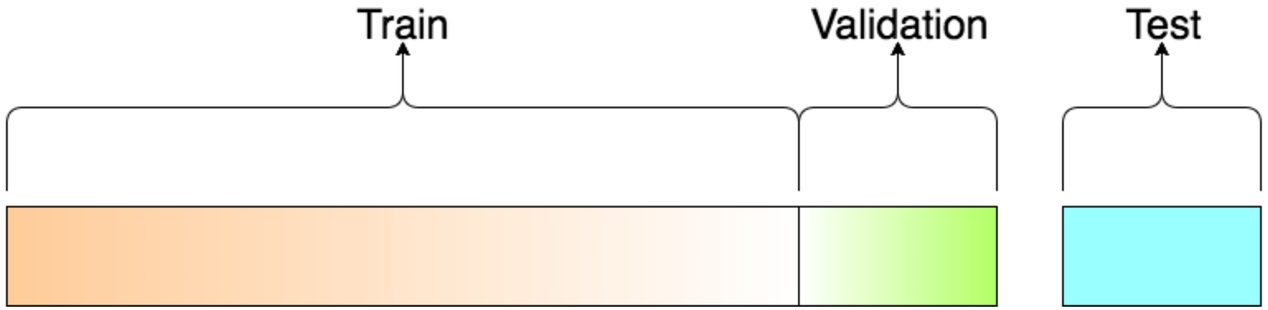
Weights

Bias

$$\hat{y} = \mathbf{w}^T \mathbf{x} + b$$

Estimate

Input features



Evaluation with the **mean squared error**

$$\text{MSE}(\mathbf{W}, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i^{(\text{test})} - \hat{y}_i^{(\text{test})} \right)^2$$

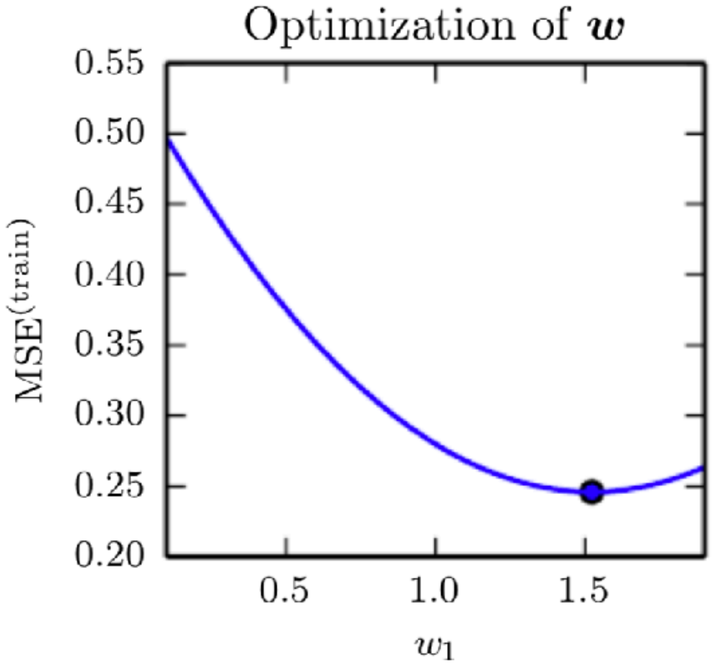


Image credits: Deep Learning (Goodfellow, Bengio & Courville); Tarang Shah

Learning: Optimizing the loss function

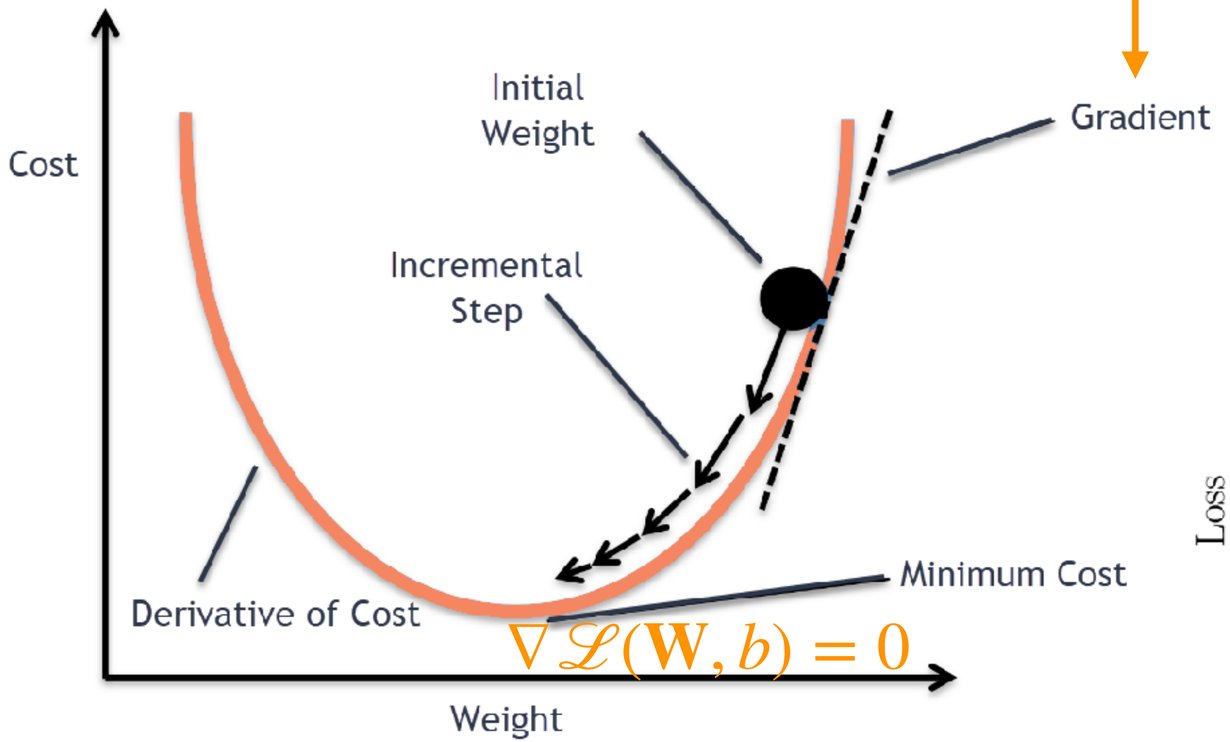
Objective: minimize the **difference** between predicted value the actual value

$$\mathscr{L}(\mathbf{W}, b) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2$$

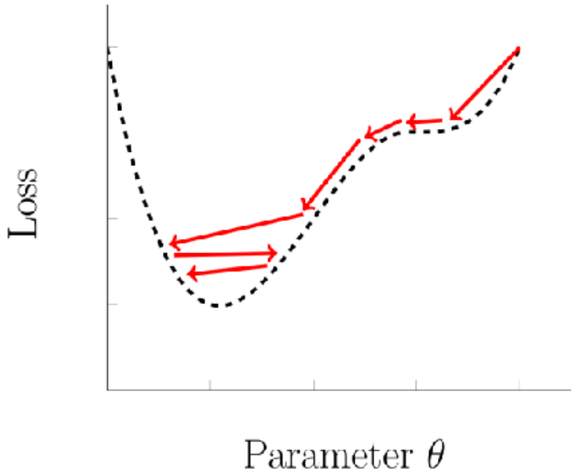
$$\mathbf{W}_{j+1} = W_j - \alpha \nabla \mathscr{L}(\mathbf{W}_j, b)$$

$\nabla \mathscr{L}(\mathbf{W}, b)$

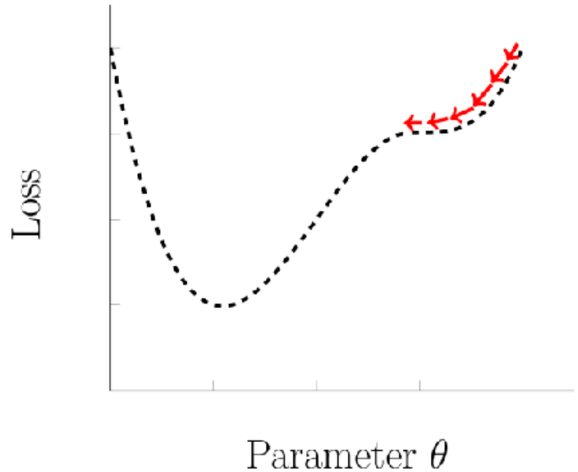
Learning rate



High Learning Rate

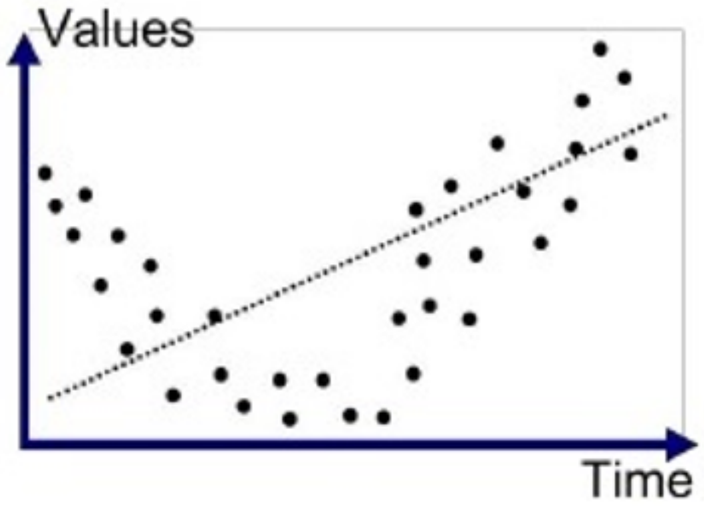


Low Learning Rate

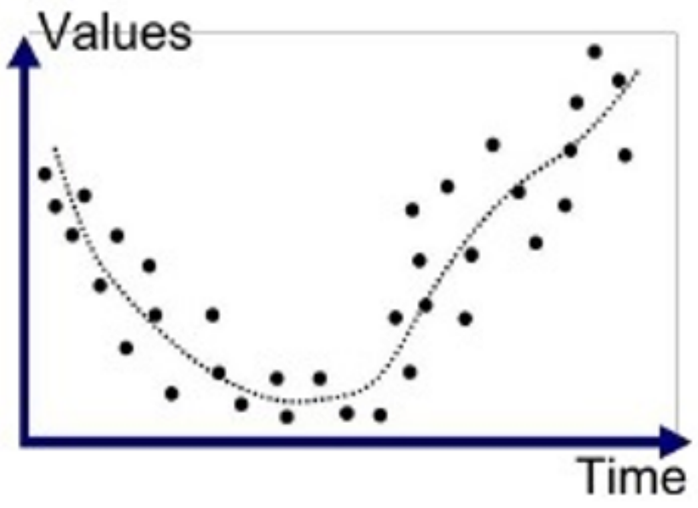


The bias-variance tradeoff

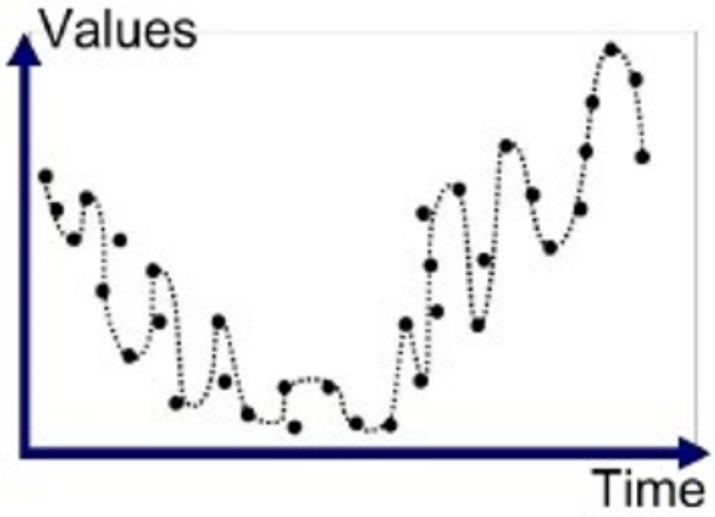
The lower the losses, the better? Not really…



Underfitted



Good Fit/Robust

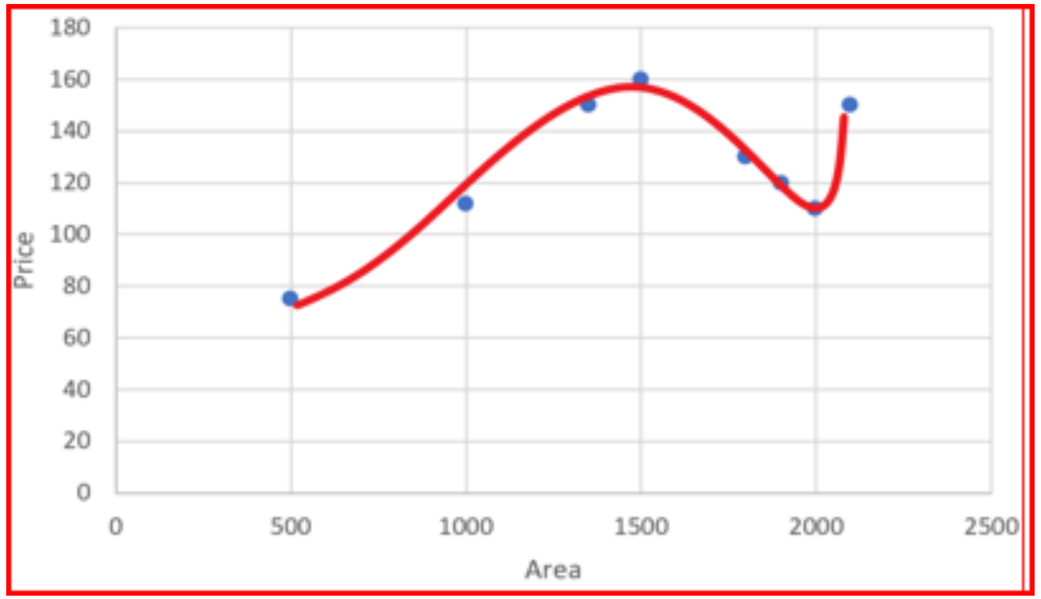


Overfitted

Regularization Technique

$$L(x,y) = \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2$$

$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 + \theta_4 x_4^4$$



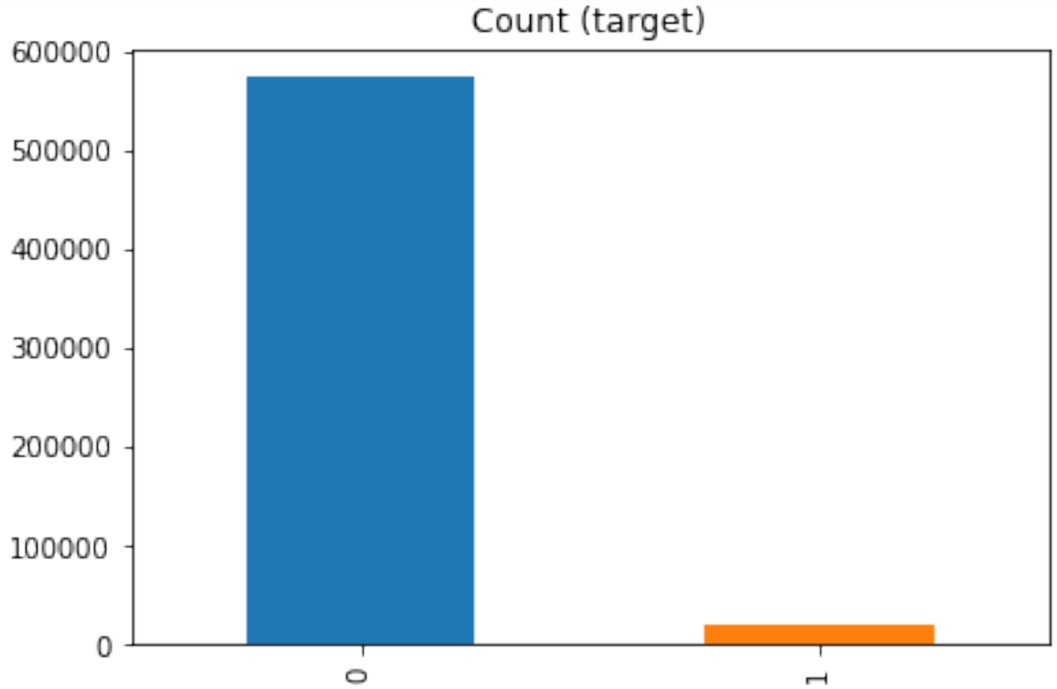
$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 + \theta_4 x_4^4$$

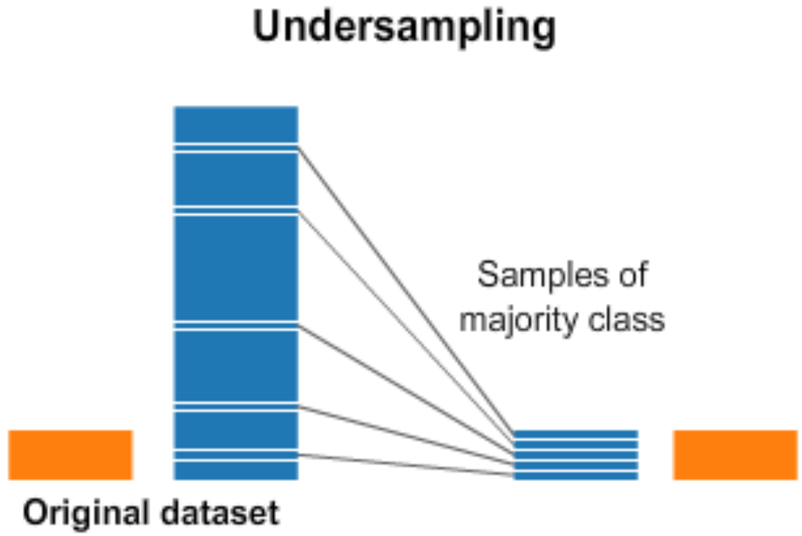
$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2$$

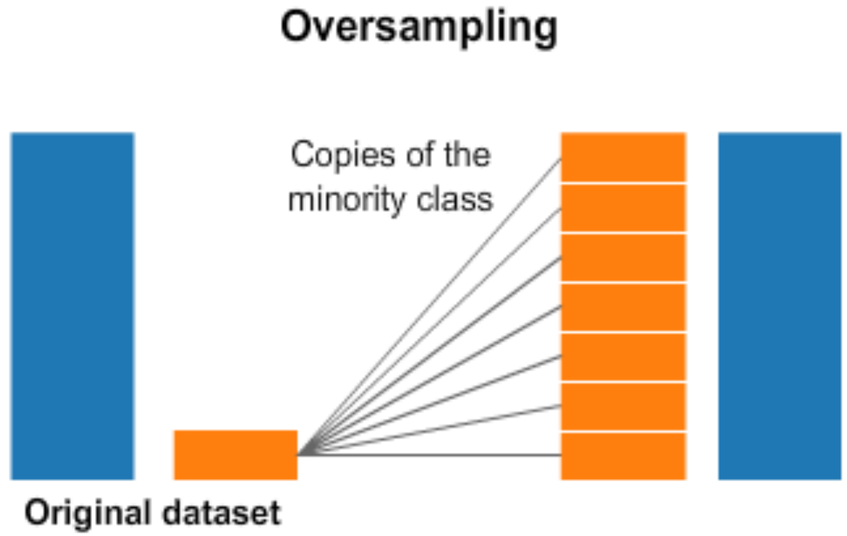
$$L(x,y) \equiv \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} |\theta_i|$$

$$L(x,y) \equiv \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} \theta_i^2$$

Balanced/Imbalanced training set



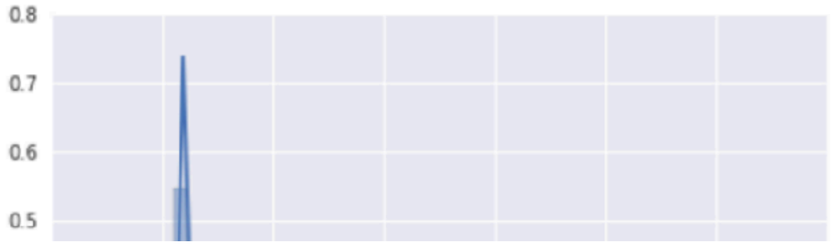


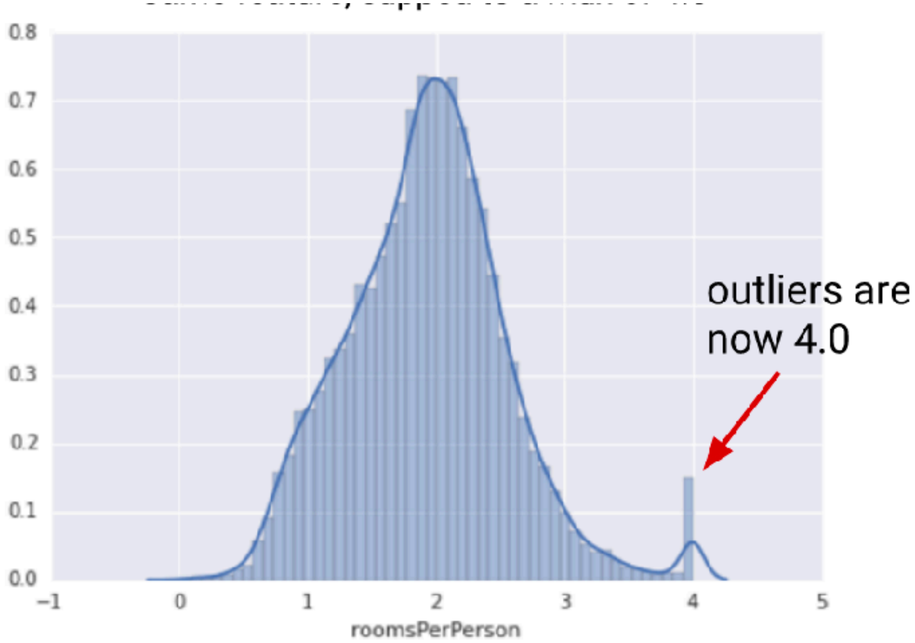


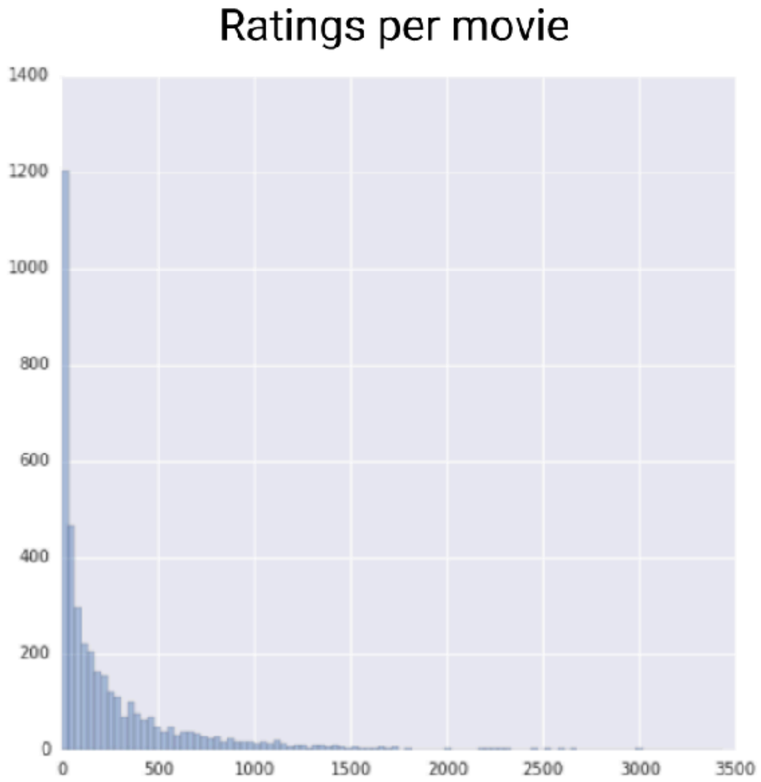
Source: kaggle.com

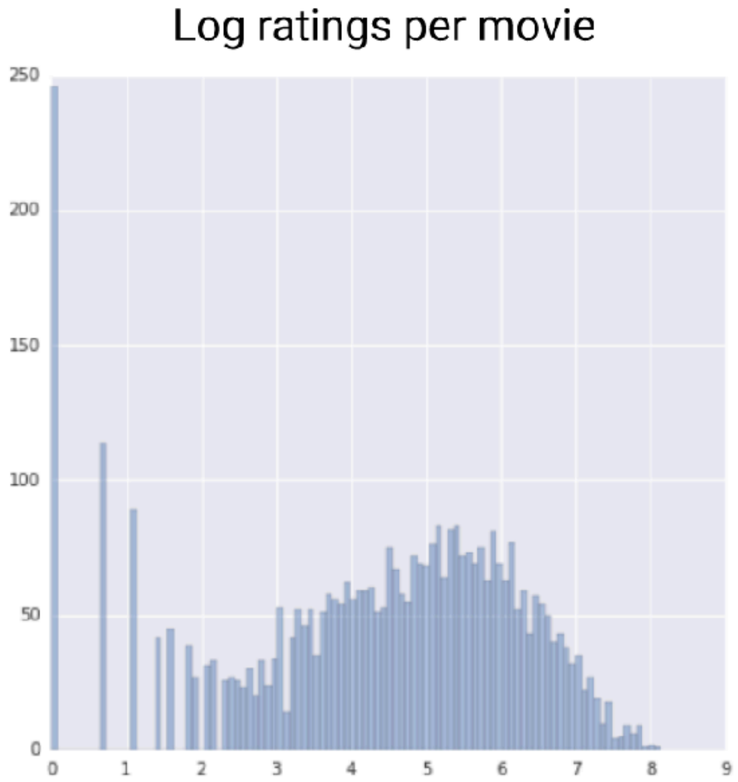
Data Normalization

A process to transform the input **data** in a **well-behaved** form.





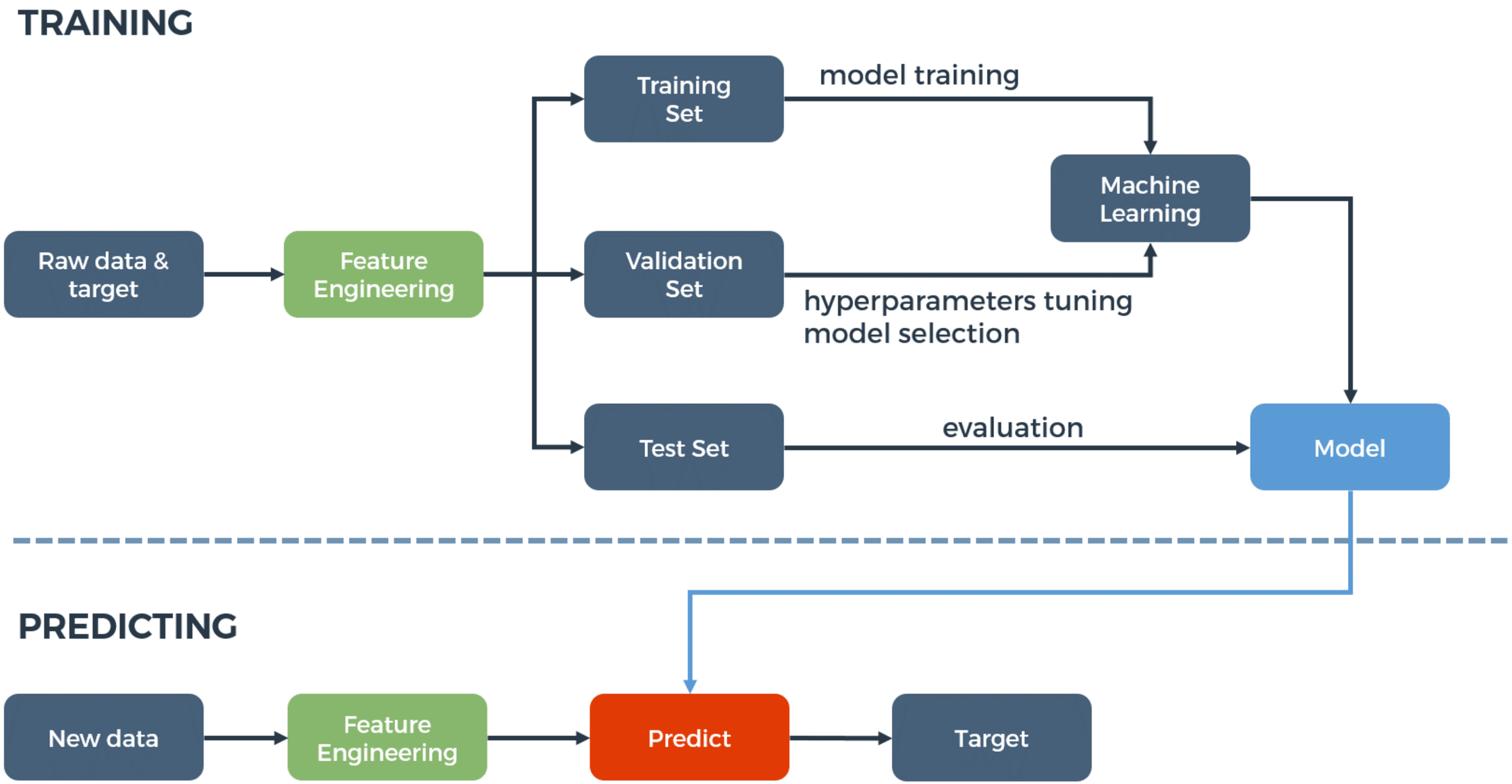




Further reading: sklearn scalers

Source: developers.google.com

General Workflow of ML/DL



Source: hackernoon.com

Categories of machine learning

Supervised

Classification
- Naive bayes
- Support vector machine
- Decision tree
- Random forest
- K-Nearest Neighbor
- Logistic regression

Regression
- Linear
- Generalized

Unsupervised

Clustering
- K-means
- K-medoids

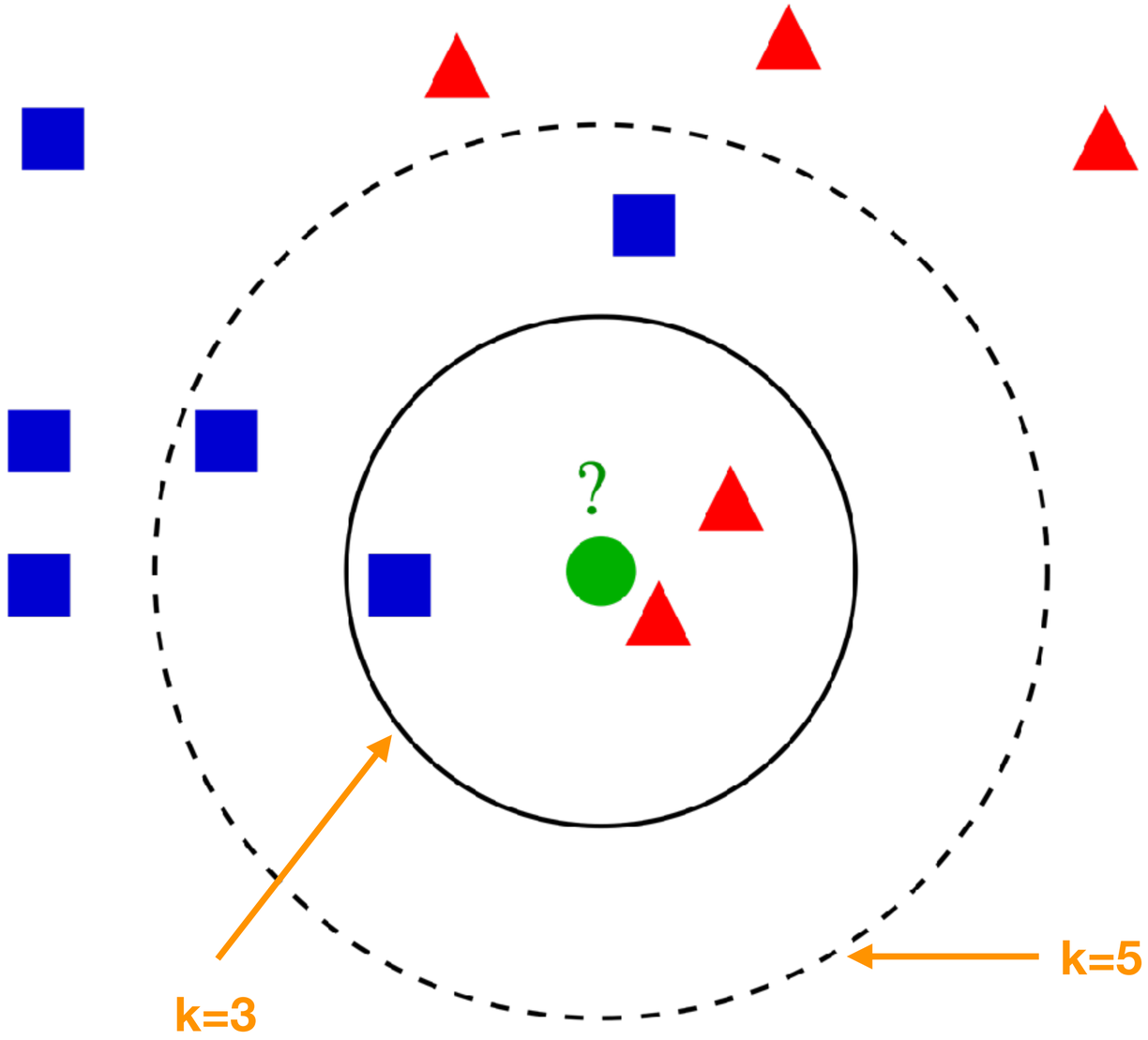
Dimensionality reduction
- PCA
- SVD

Reinforcement

Discrete
- Markov decision process
- Deep Q Network
- A2C
- A3C

Continuous
- DDPG
- NAF

k-Nearest Neighbor

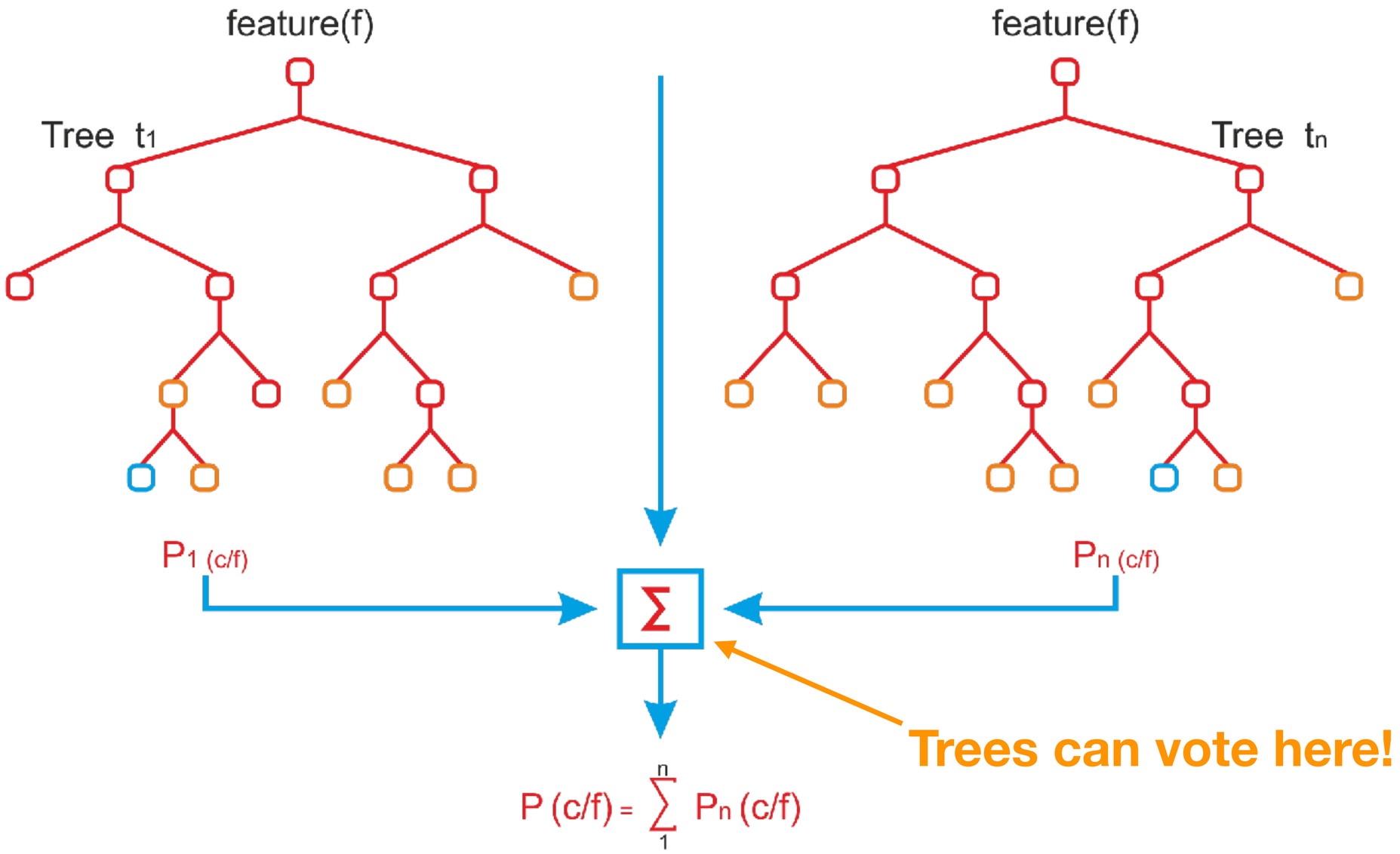


… is a **classification** algorithm

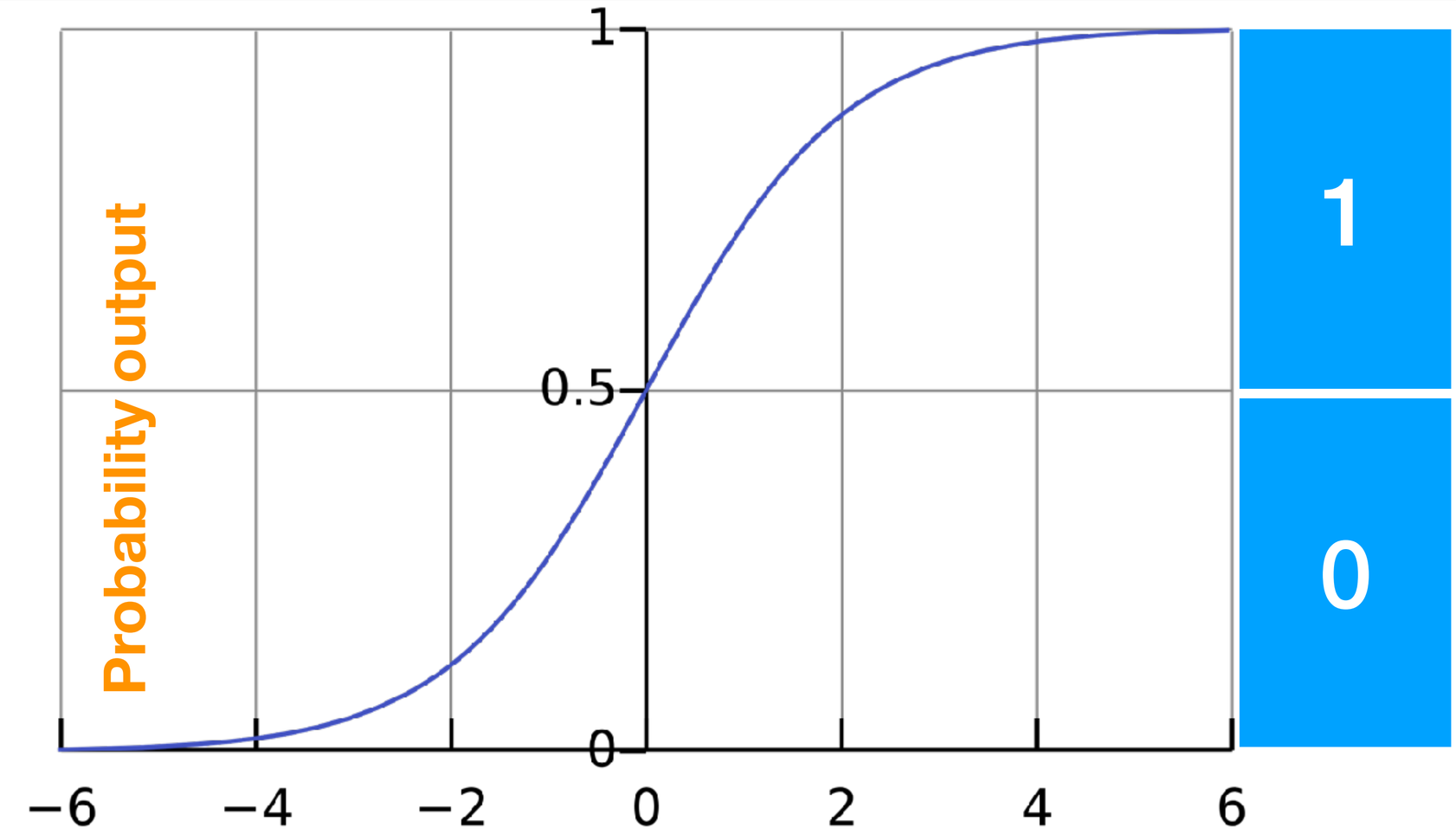
… based on a **voting** process

Image source: Wikipedia

Decision Tree → Random Forest



Logistic Regression



Logistic sigmoid function

$$\phi(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}}}$$

… is a binary **classification** algorithm, not a regression algorithm

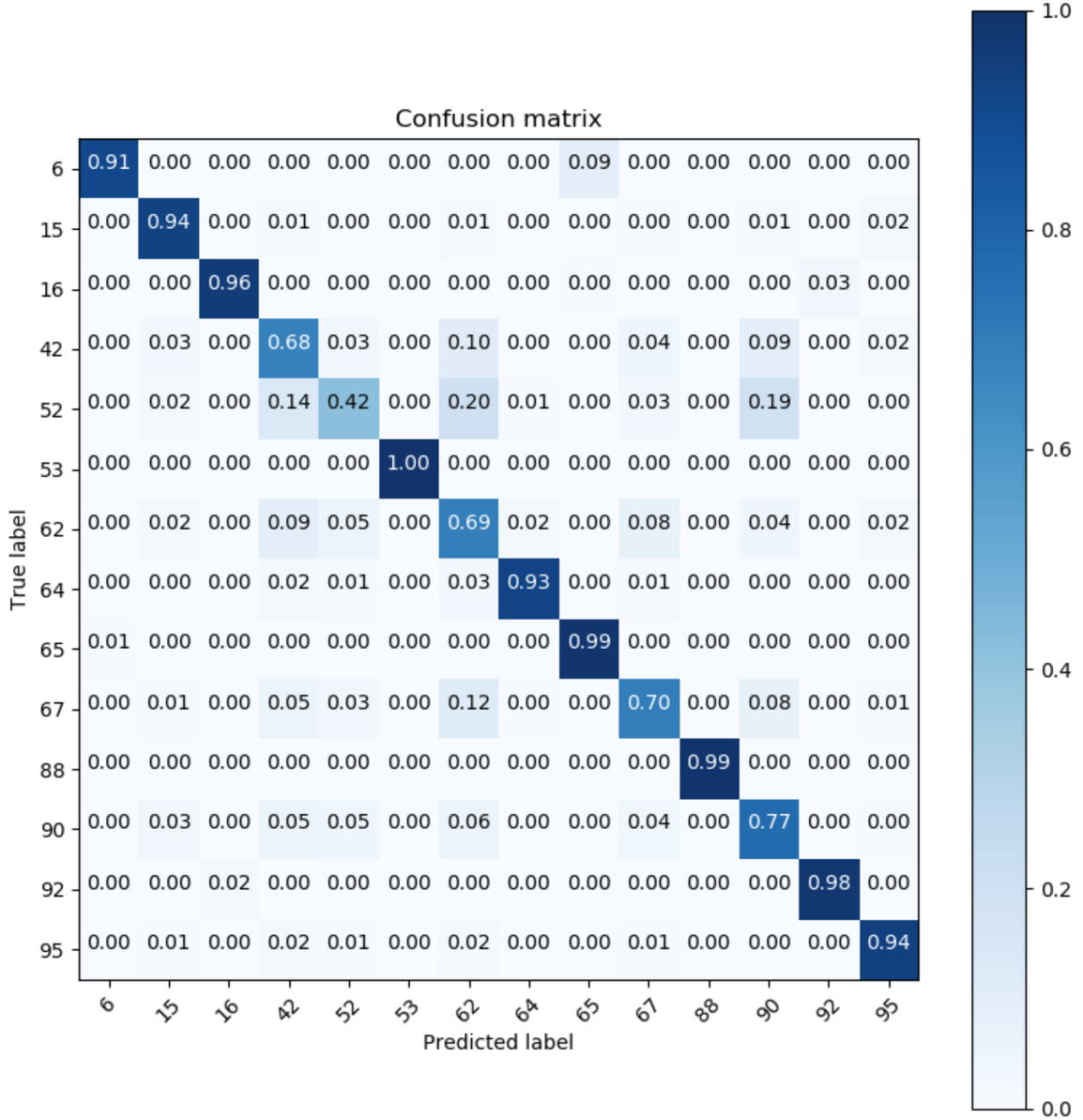
… Gives the **probability** of each class

Confusion Matrix

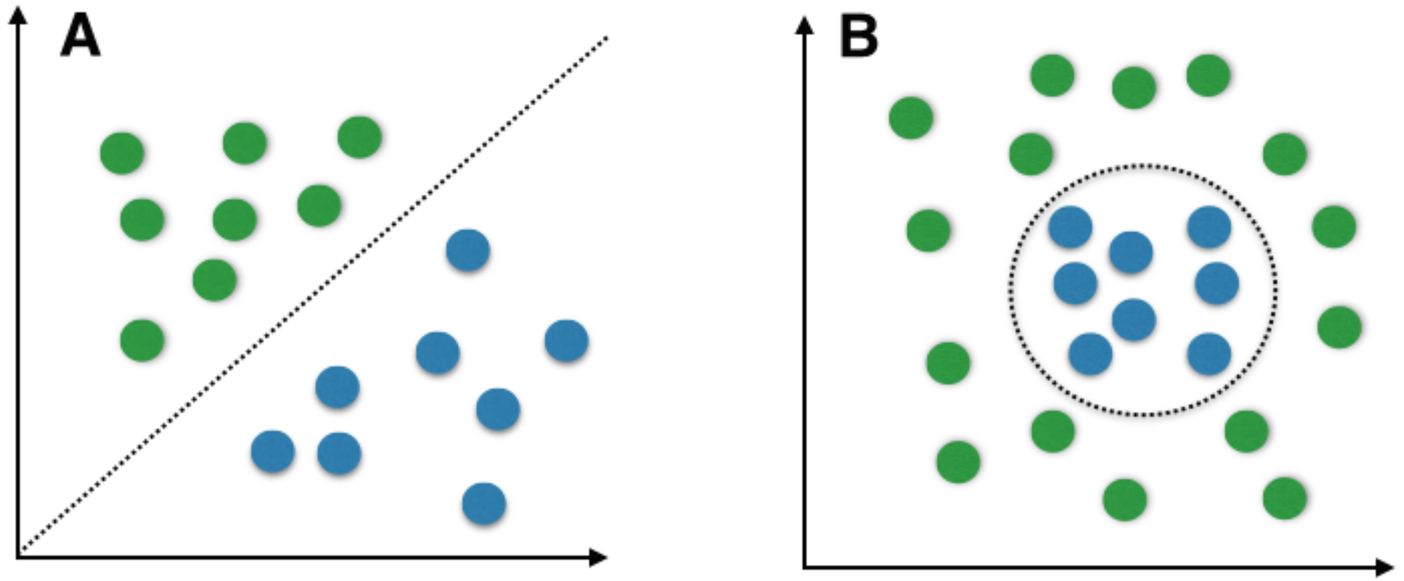
		Prediction outcome		
		p	n	total
actual value	p′	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

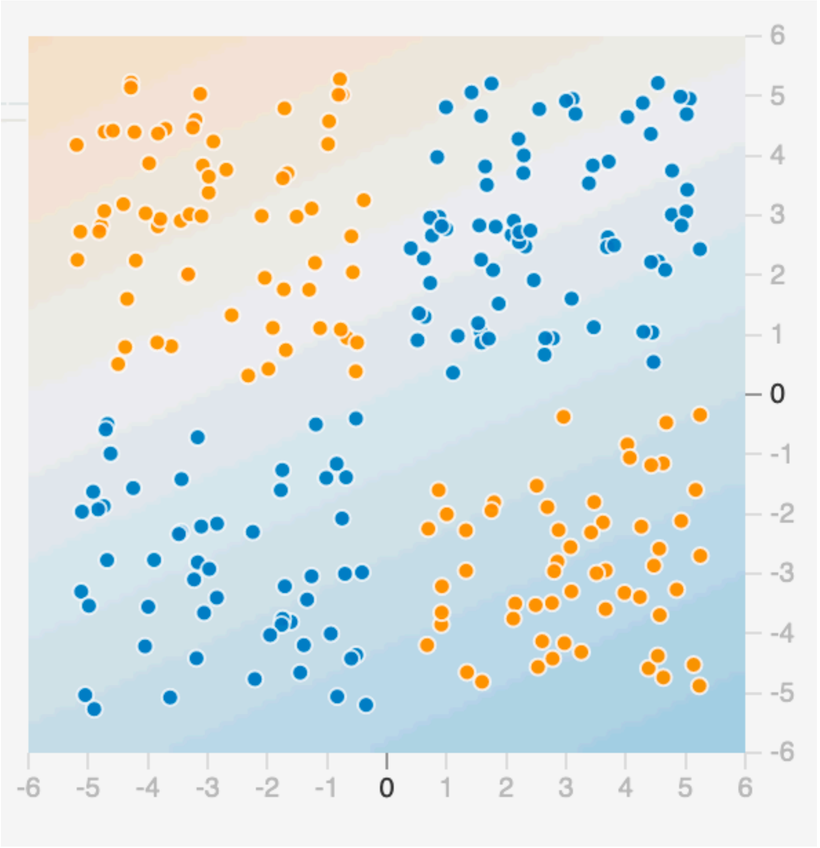
Accuracy = (TP + TN) / (TP + FN + FP + TN)
Precision (p) = TP / (TP + FP)
Recall (r) = TP / (TP + FN)

$$F_1 = \frac{2}{r^{-1} + p^{-1}}$$



What if the problem is not so simple?

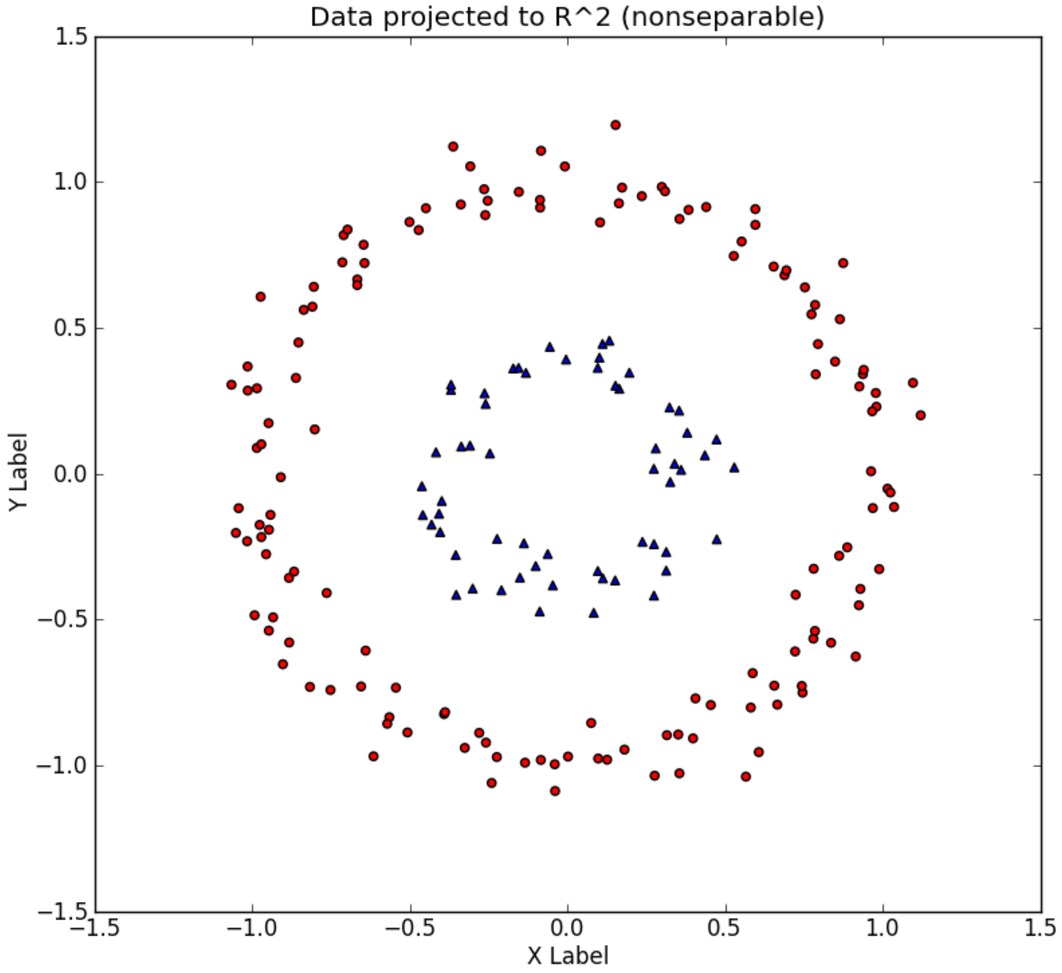


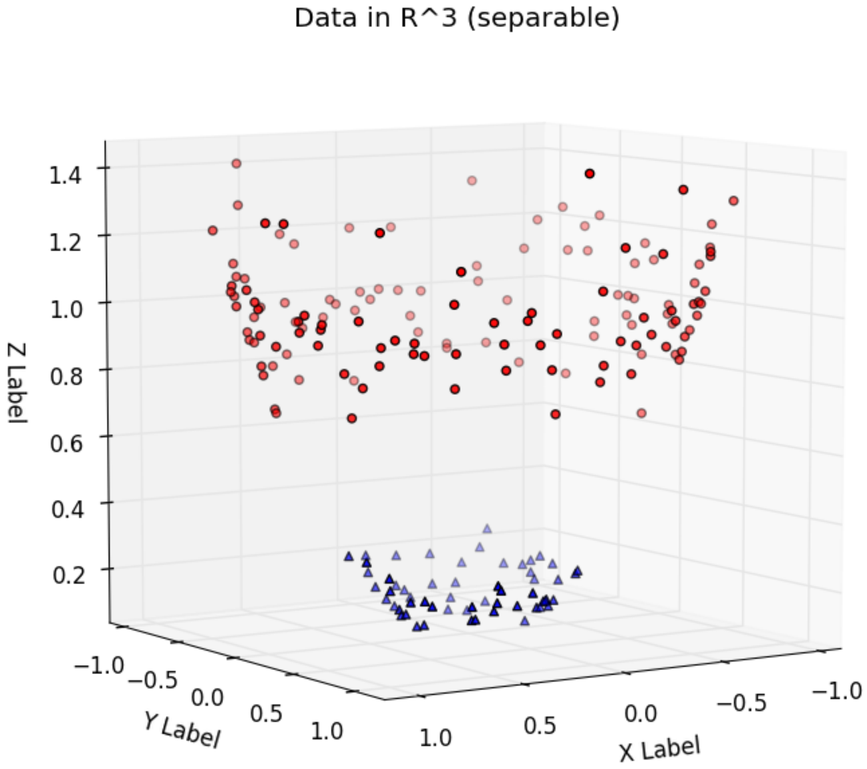


Possible solutions:

- map into another space (kernel trick)
- Add more layers (deep learning)

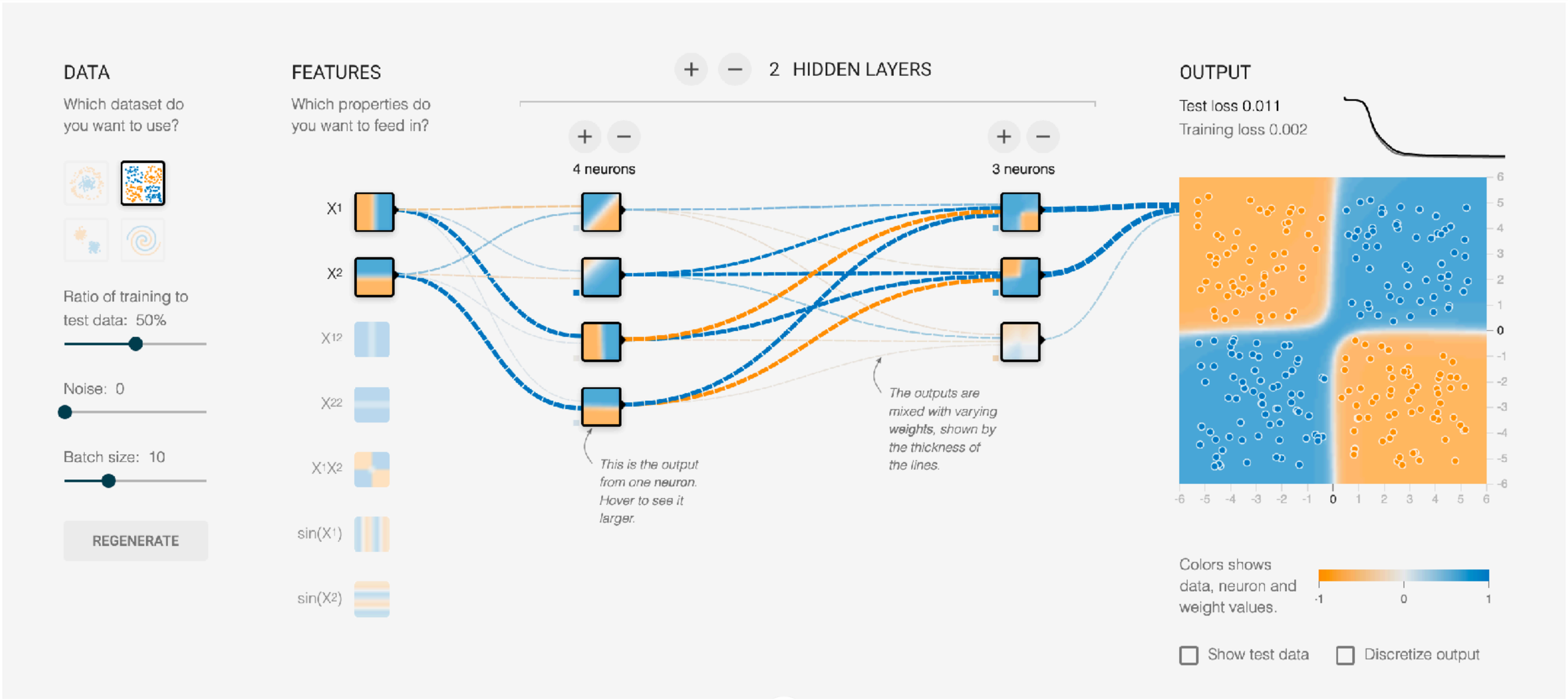
Kernel Trick



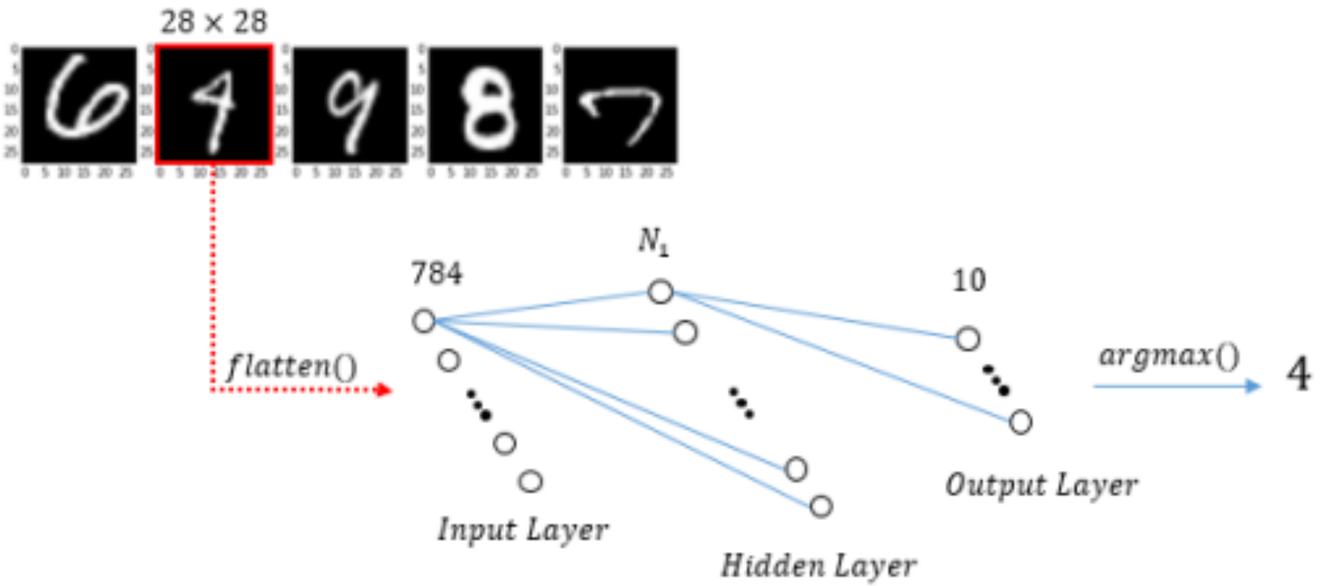


Or, use a deeper neural network…

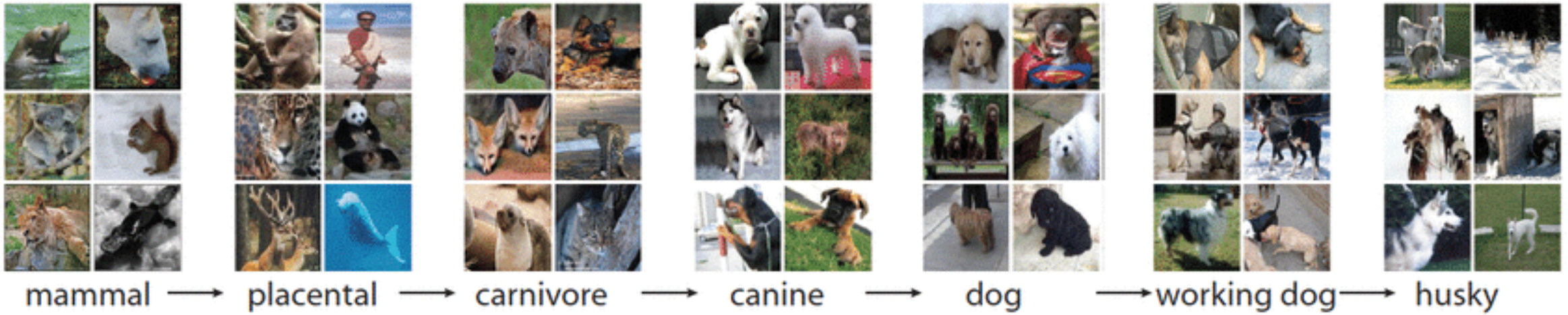
Live demo: Multi-layer Perceptrons

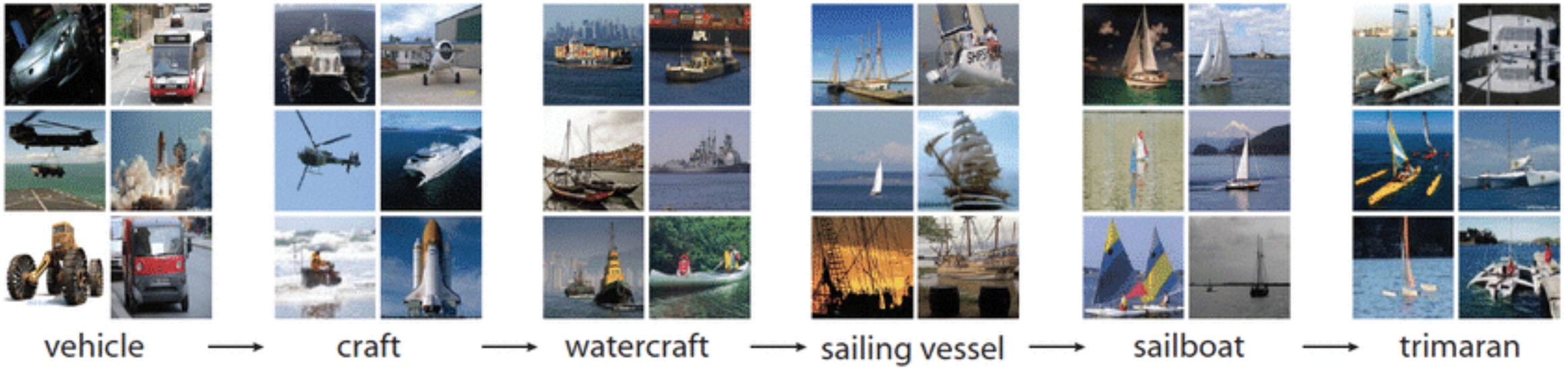


https://playground.tensorflow.org


28 × 28
784
N_1
10
flatten()
argmax()
4
Input Layer
Hidden Layer
Output Layer

What about these images? Should we make the neural network as deep as possible?





ImageNet dataset

Deep Neural Network ==> Deep Learning

What about these images? Should we make the neural network as deep as possible?

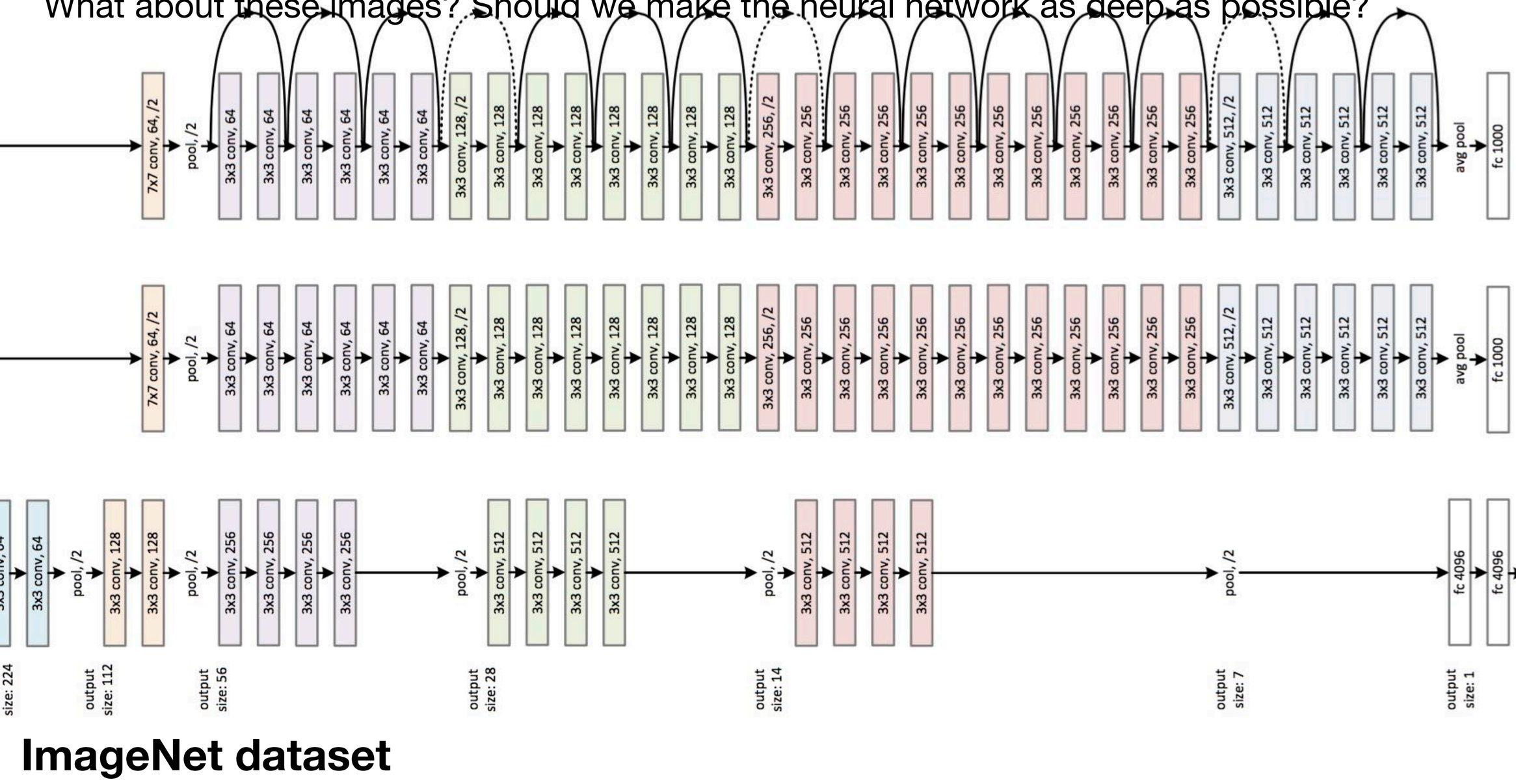
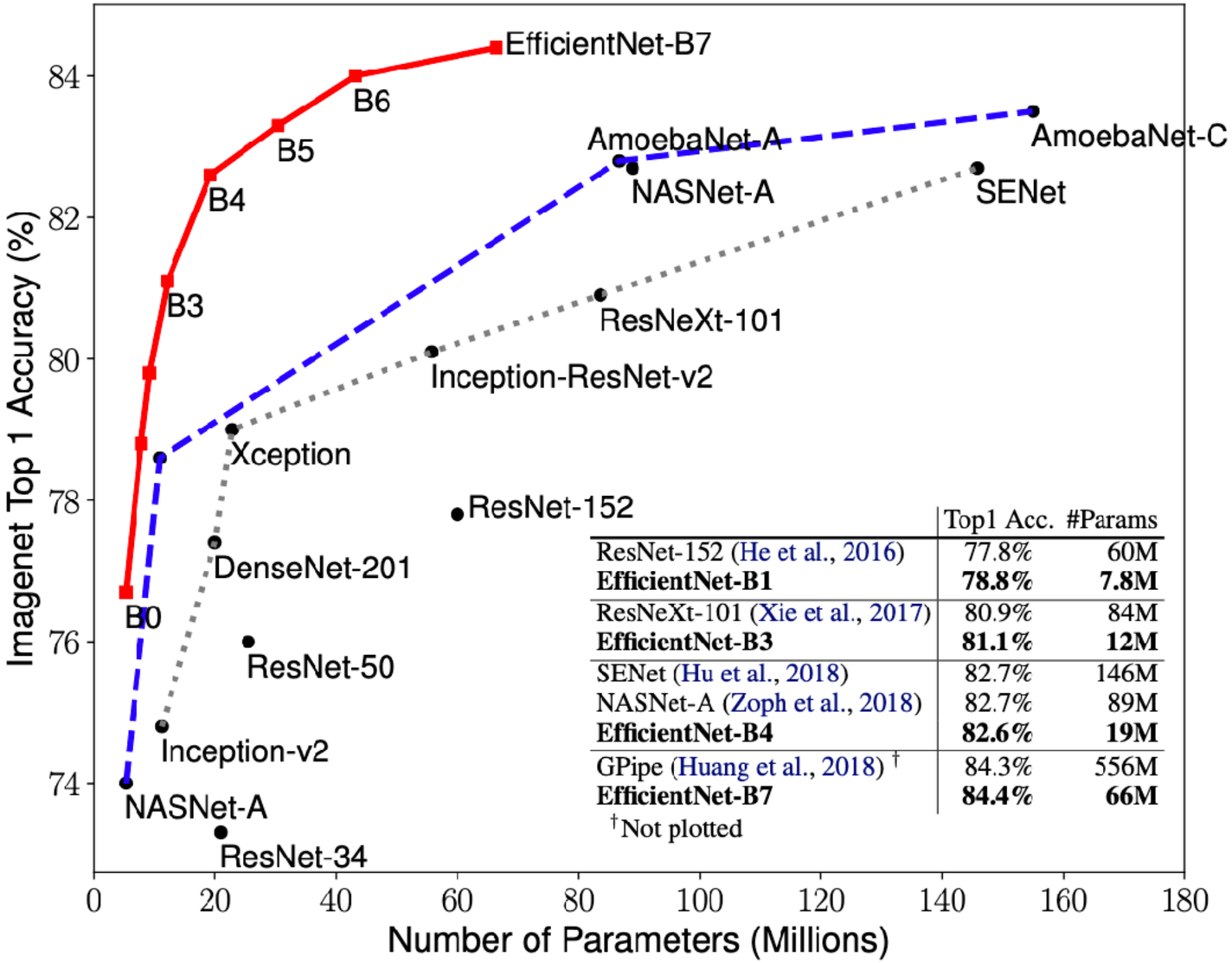


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Mid-**

ImageNet dataset

Deep Neural Network ==> Deep Learning

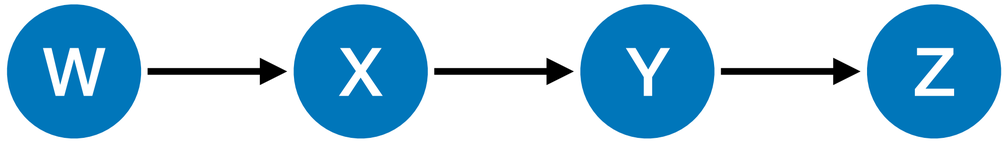
Tan & Le (2019, arXiv:1905.11946)



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

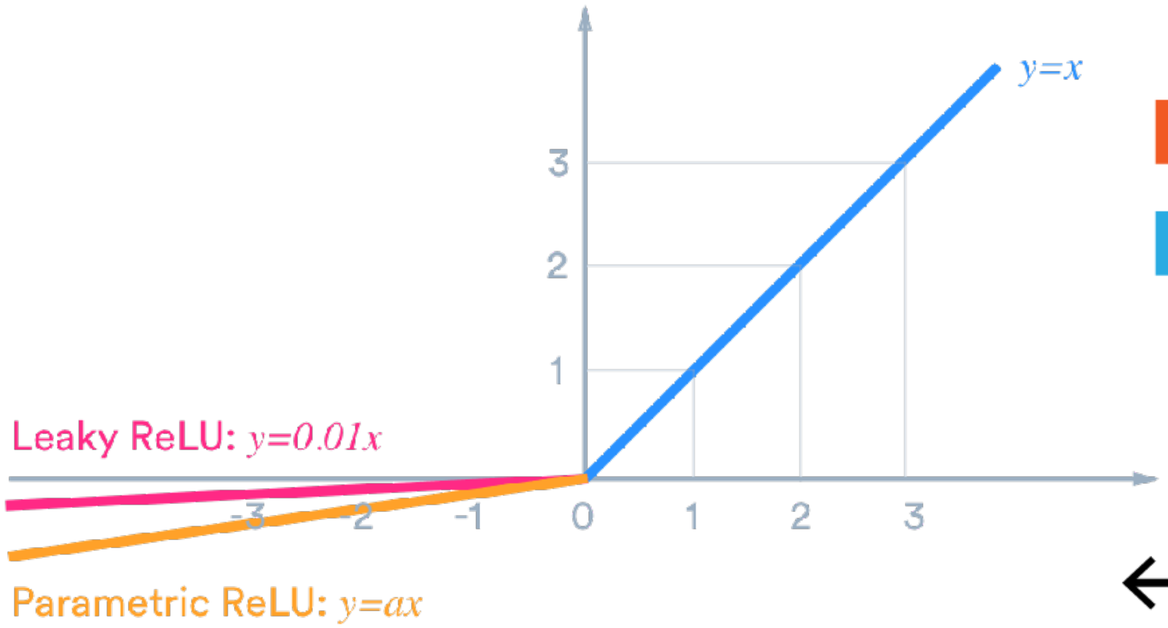
†Not plotted

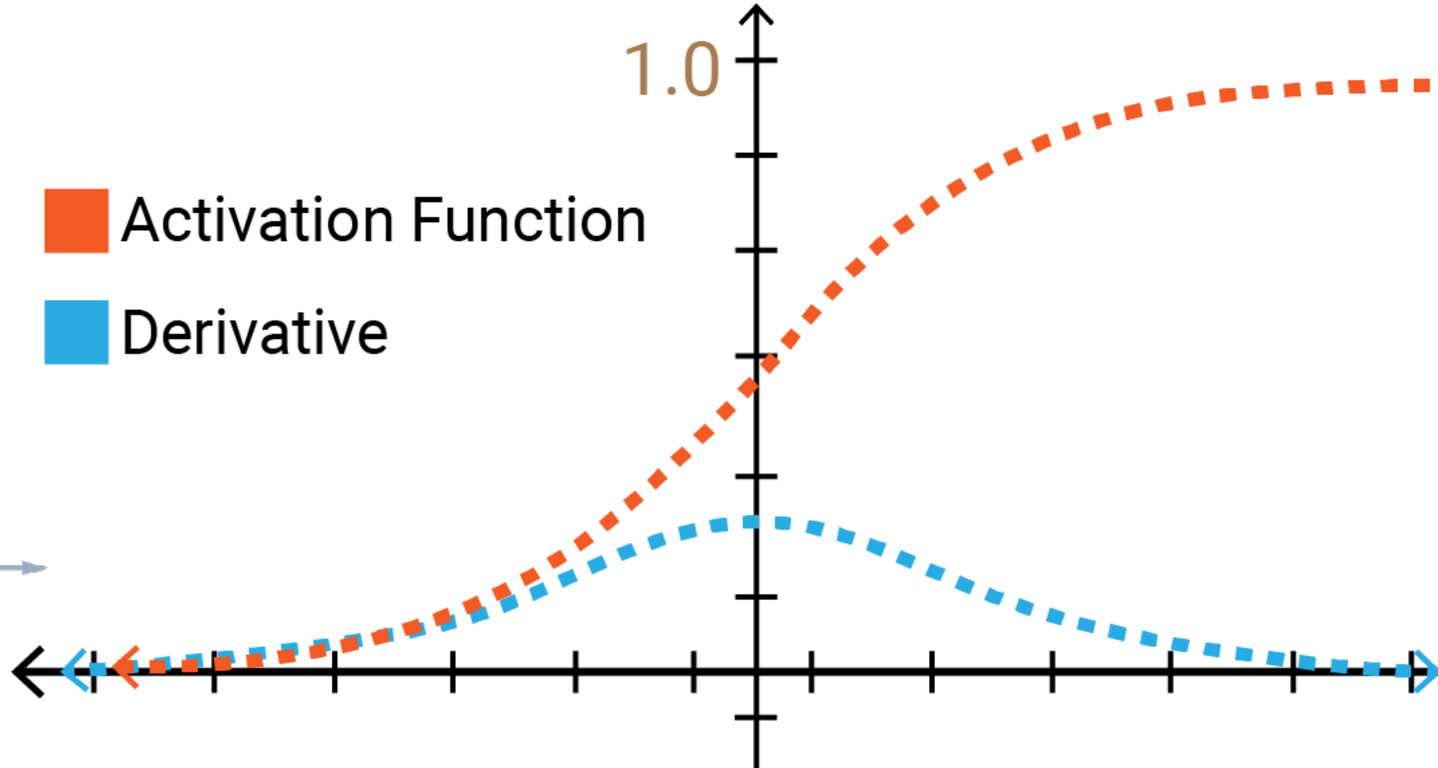
The vanishing gradient problem



Chain rule

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w}$$

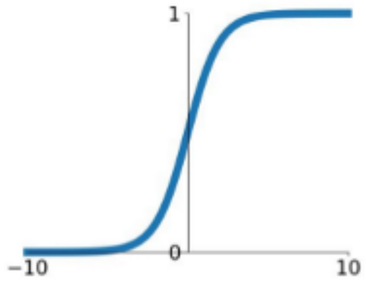




Activation Functions

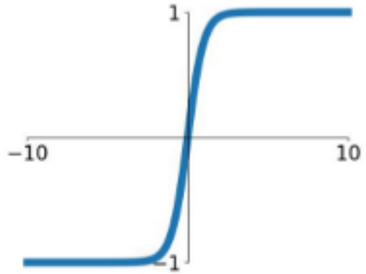
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



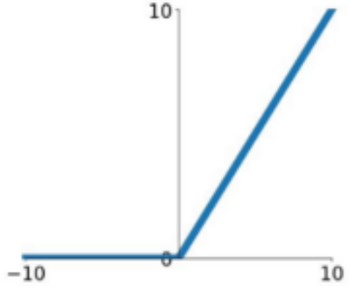
tanh

$$\tanh(x)$$



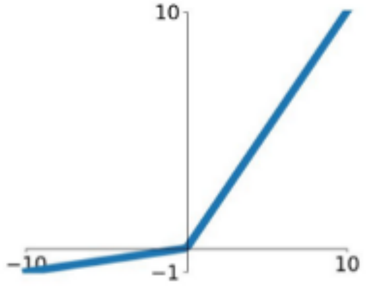
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

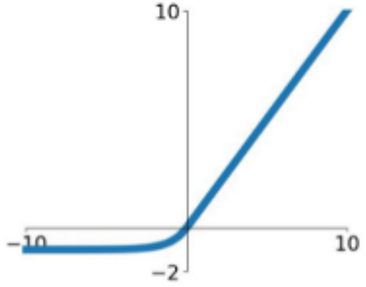


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

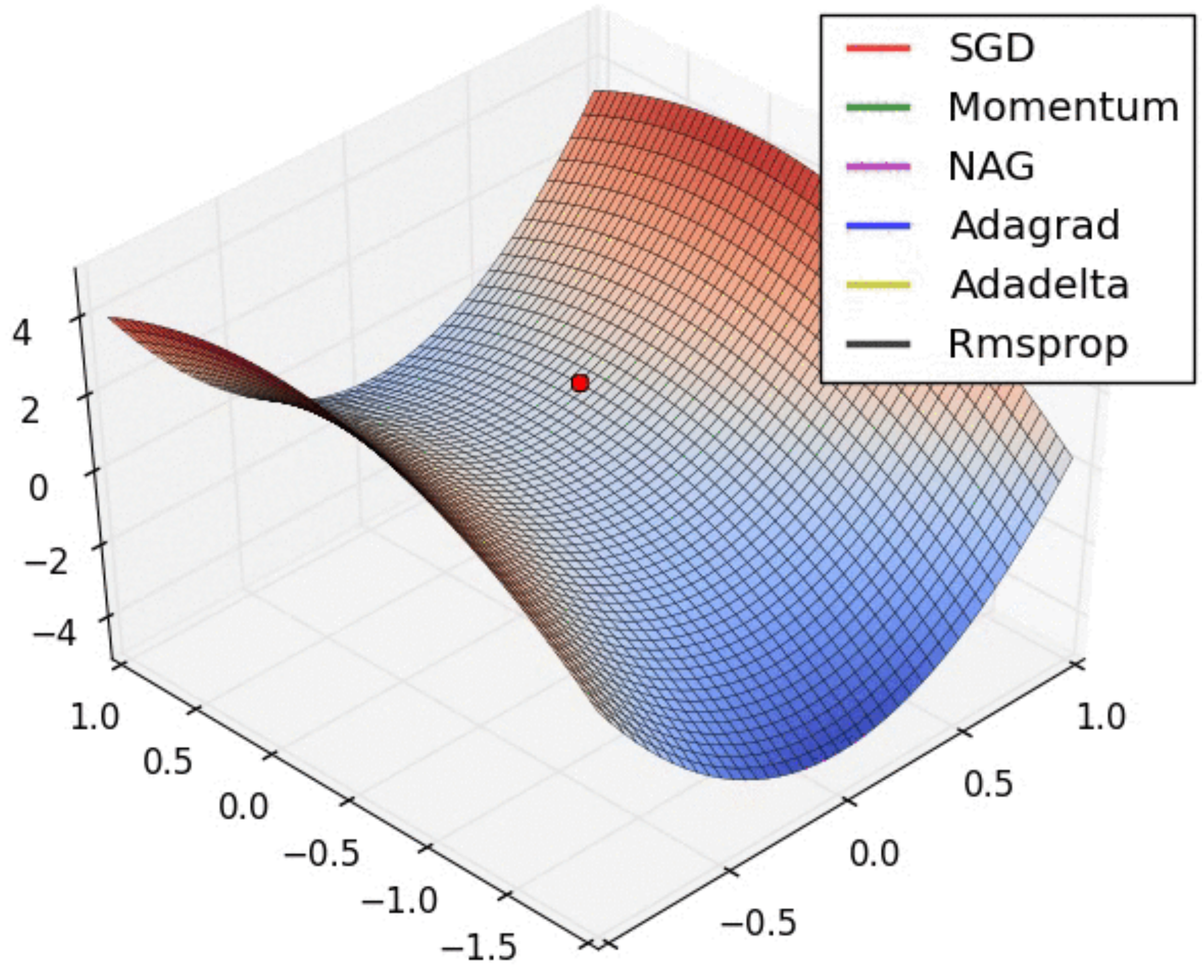
ELU

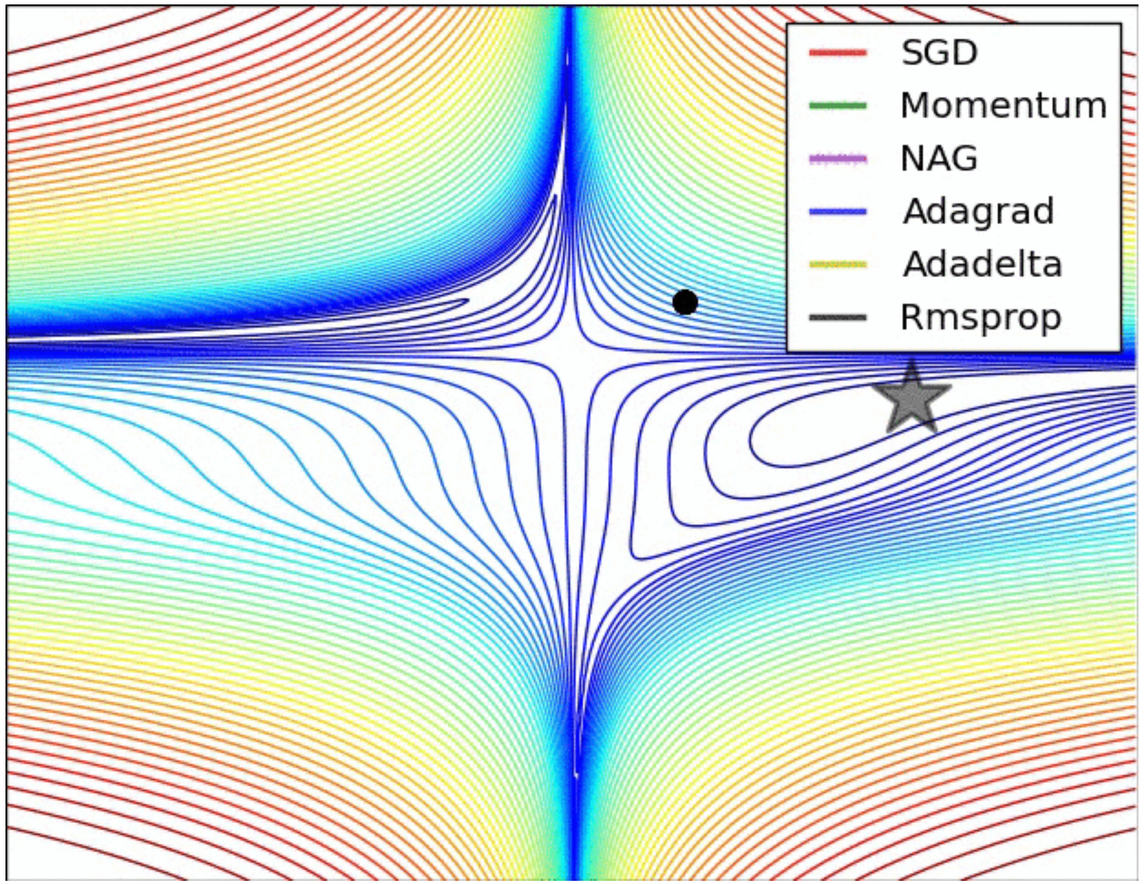
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



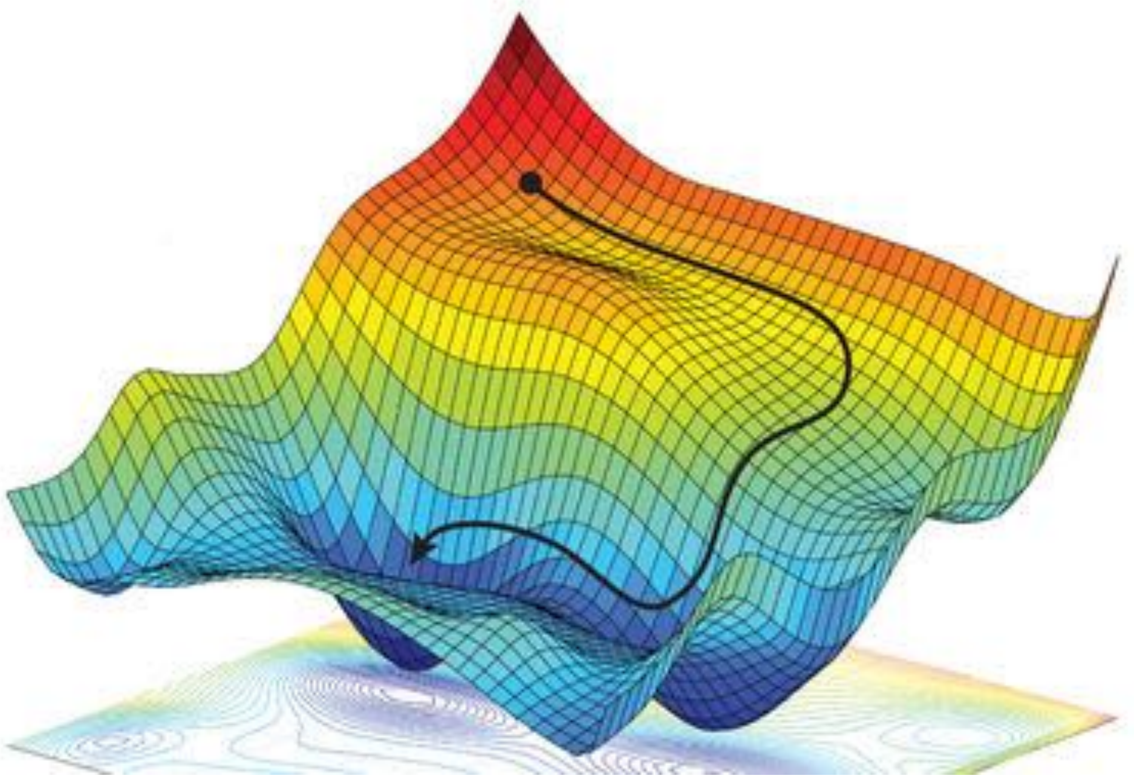
But one can design their own activation functions!

Optimizers





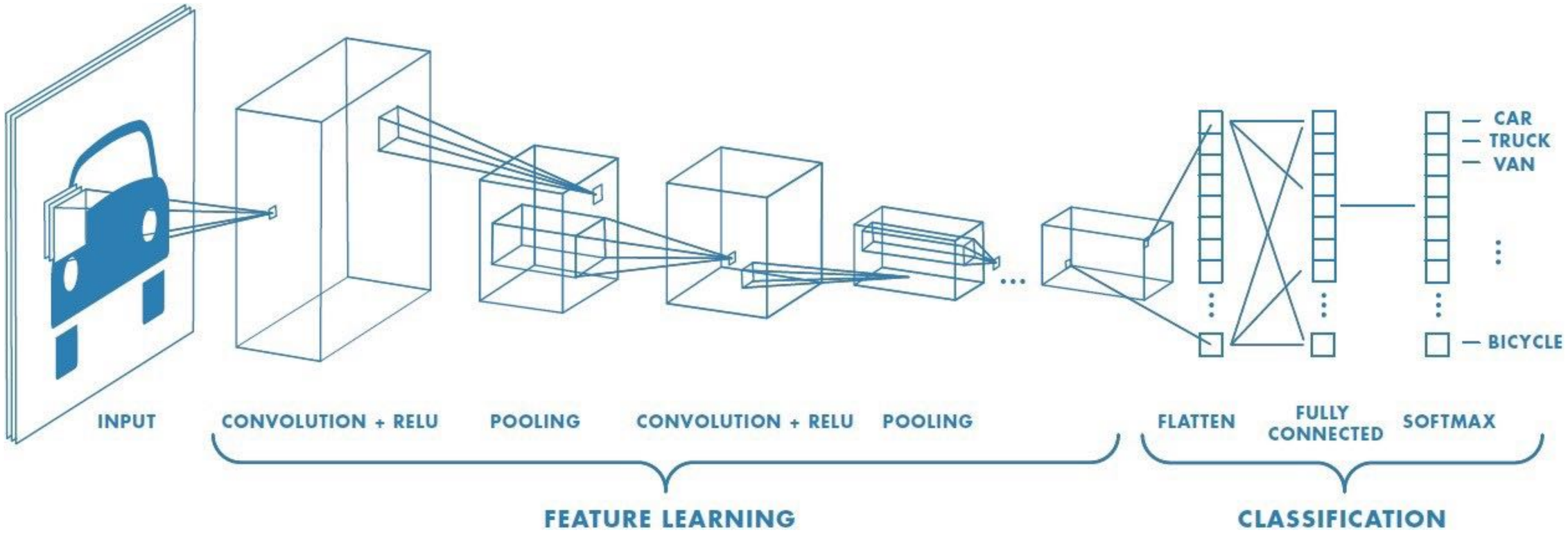
Source: medium.com/analytics-vidhya



Building blocks of DNN

CNN, Dense, LSTM, etc.

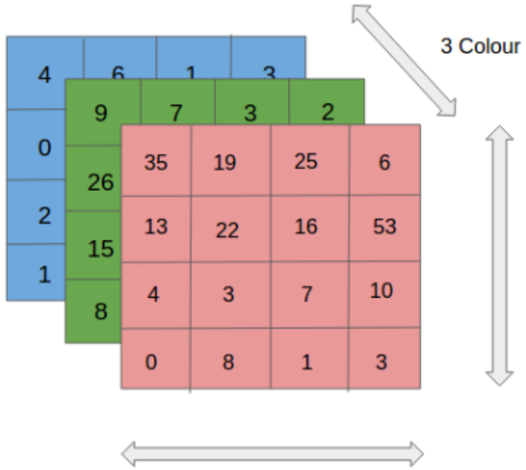
Convolution Neural Network (CNN)

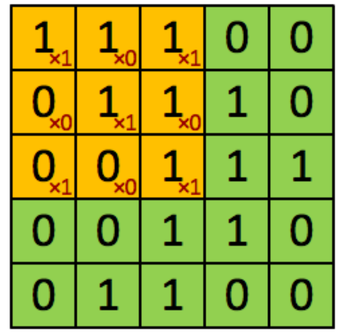


CNN: Inspired by the organization of visual cortex, which responds to stimuli only in a restricted region

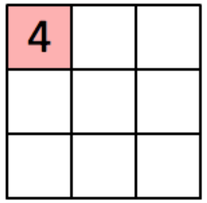
Image credit: Sumit Saha

Convolution Neural Network (CNN)

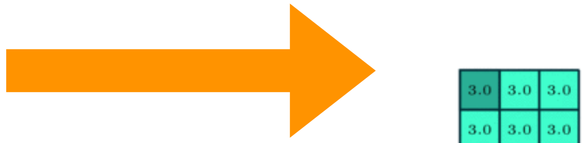


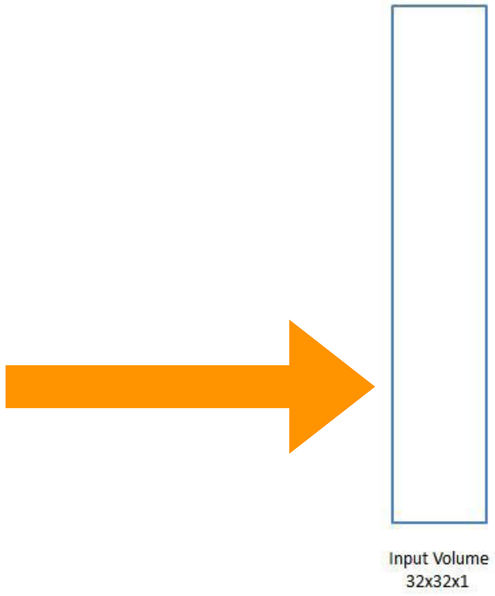


Image

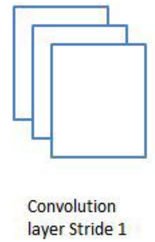


Convolved Feature





Input Volume
32x32x1

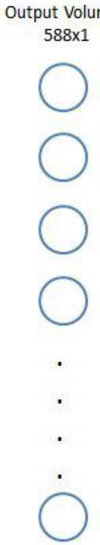


ReLU Activation Fn.
Volume-28x28x3

Convolution
layer Stride 1

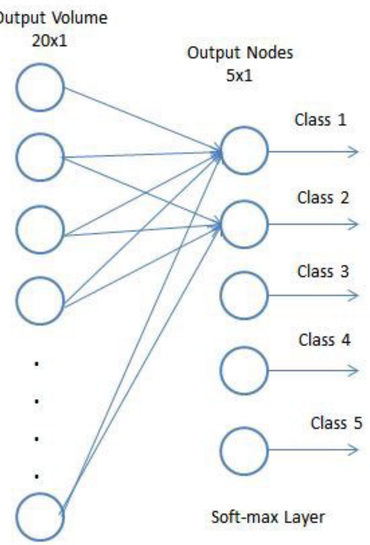
Output Volume
14x14x3

Max Pool
layer Stride 2



Output Volume
588x1

Flatten layer



Output Volume
20x1

Fully connected
Layer ReLU Activation
Fn.

Output Nodes
5x1

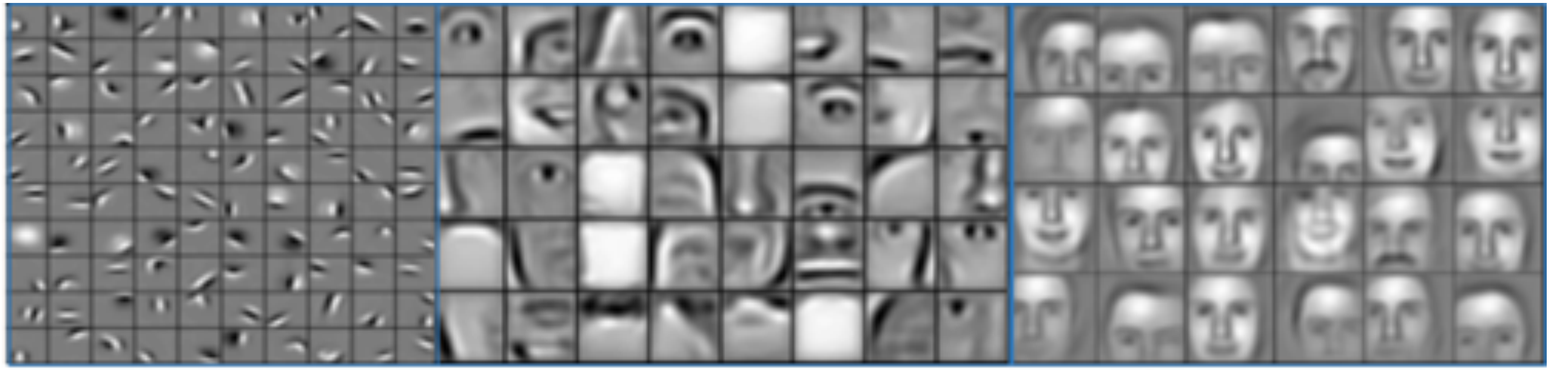
Class 1
Class 2
Class 3
Class 4
Class 5

Soft-max Layer

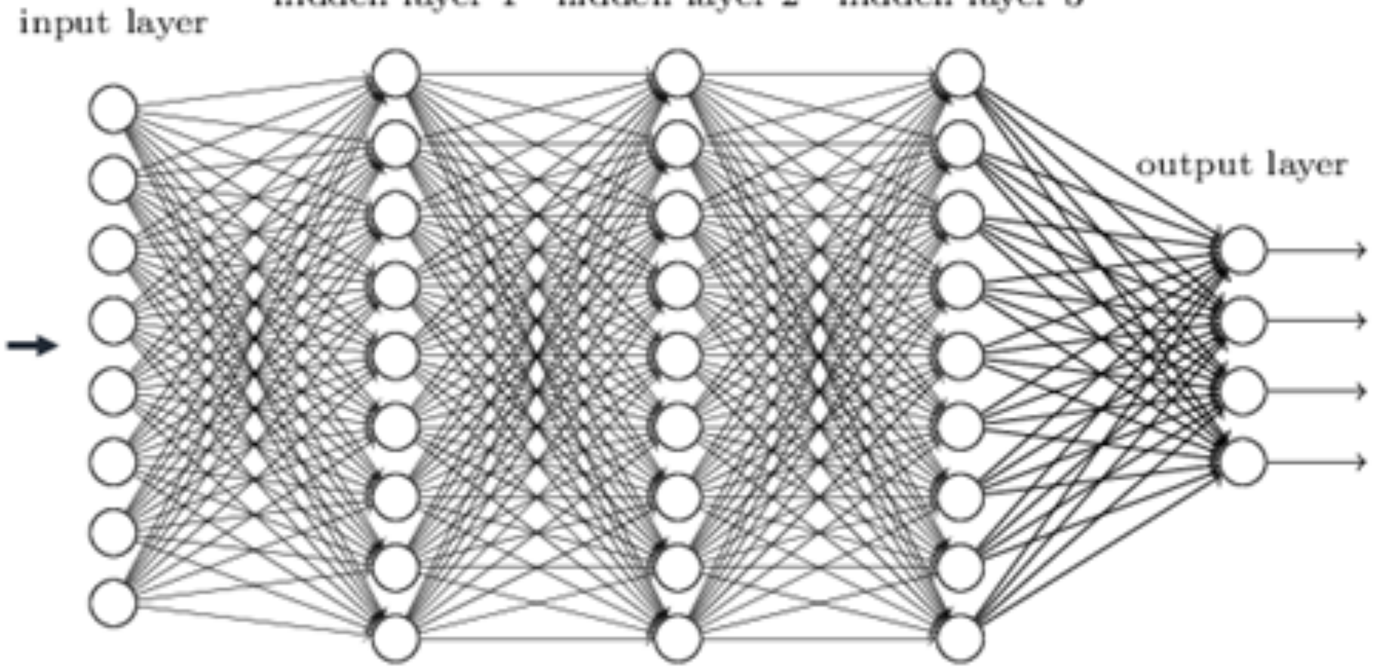
Soft-Max
Activation Fn

CNNs are hierarchical

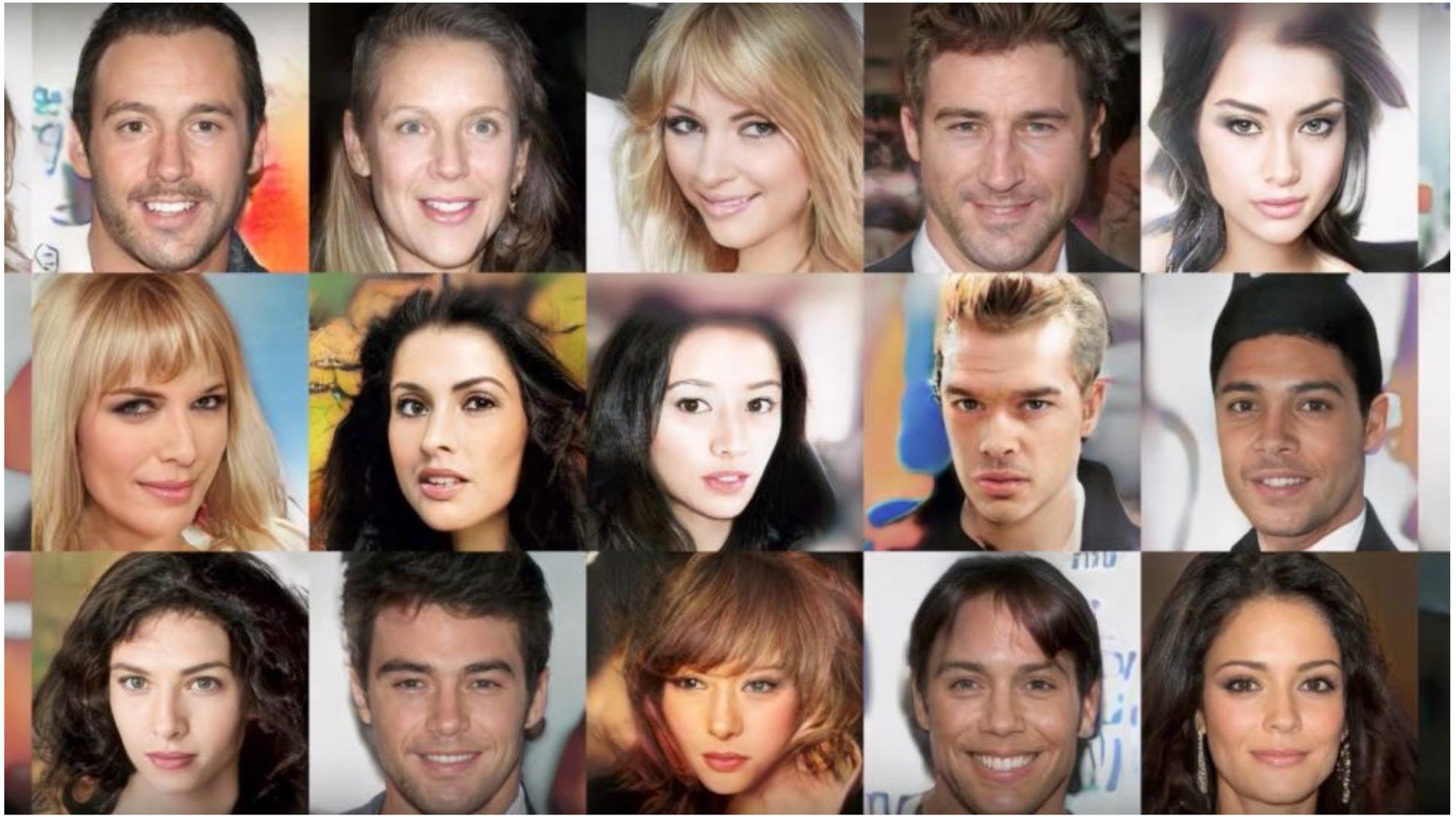
Deep neural networks learn hierarchical feature representations



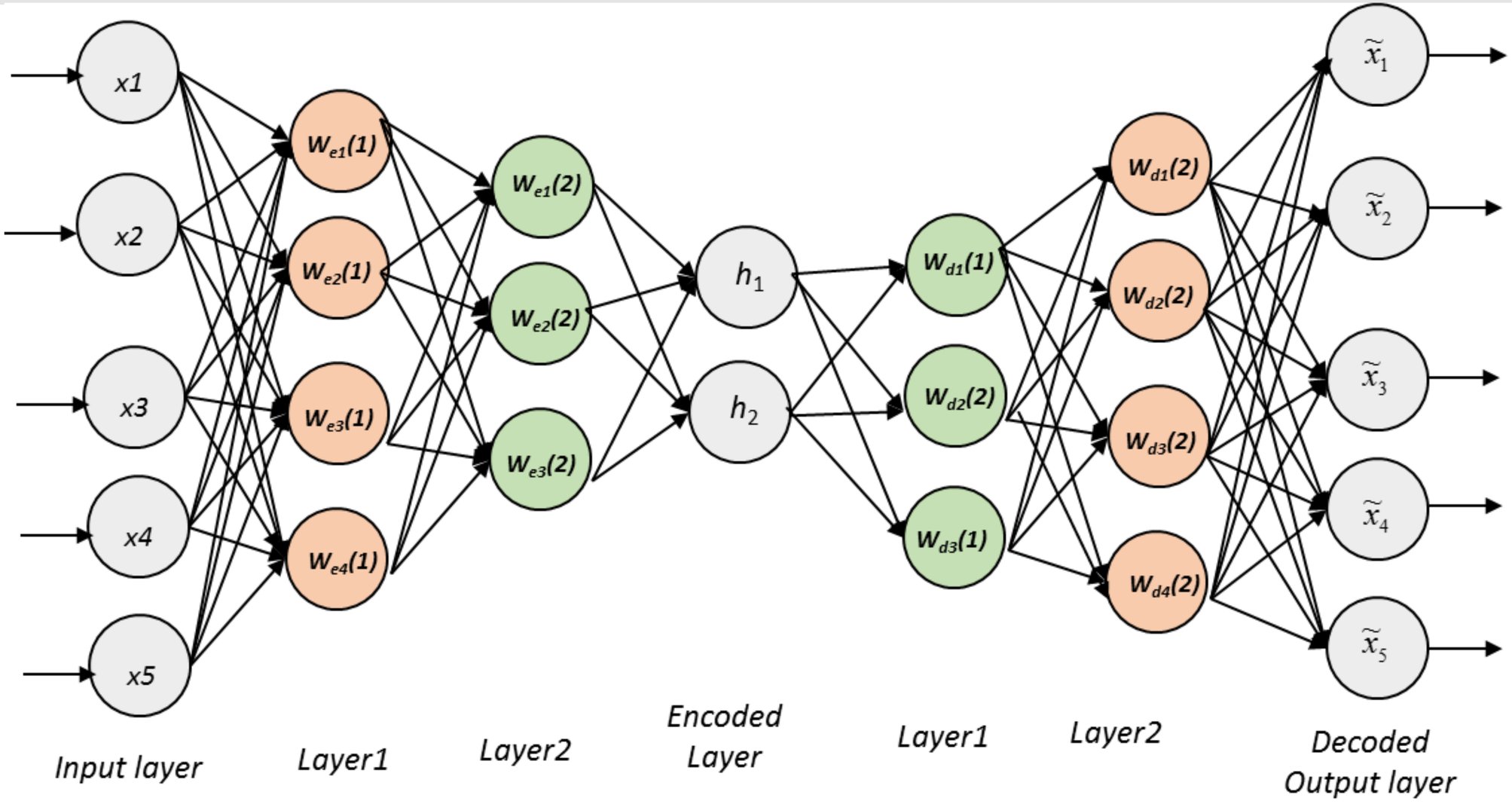
hidden layer 1 hidden layer 2 hidden layer 3



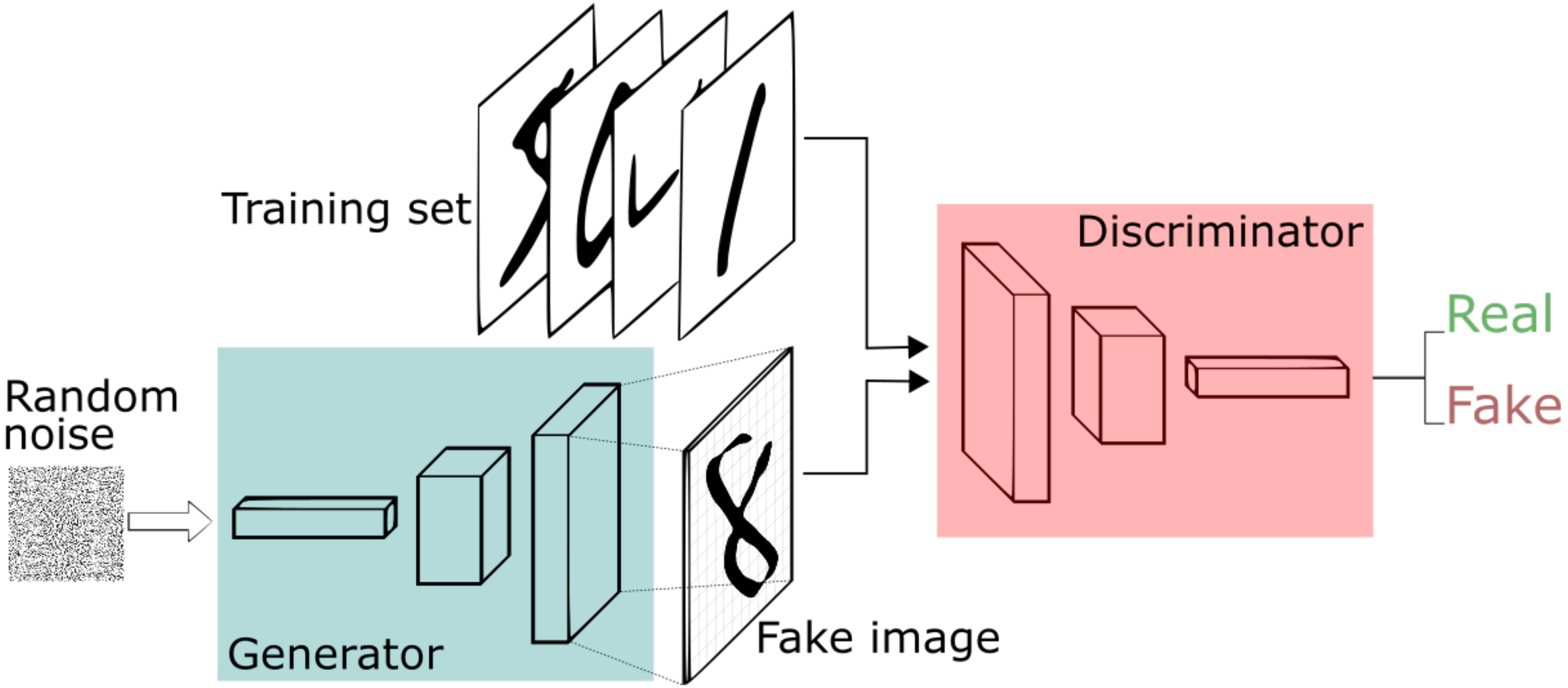
How about generating images from labels?



Autoencoders



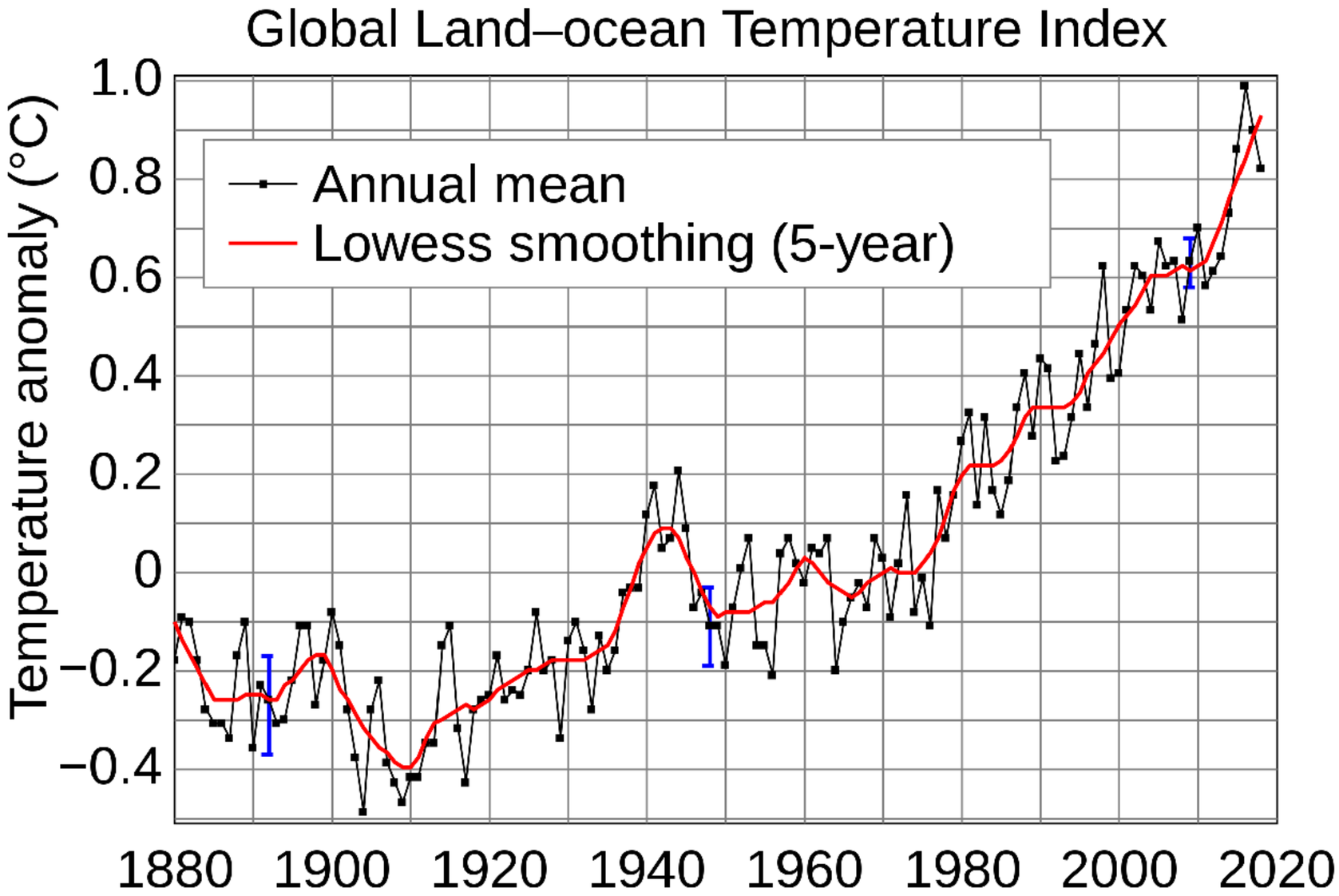
Generative Adversarial Networks



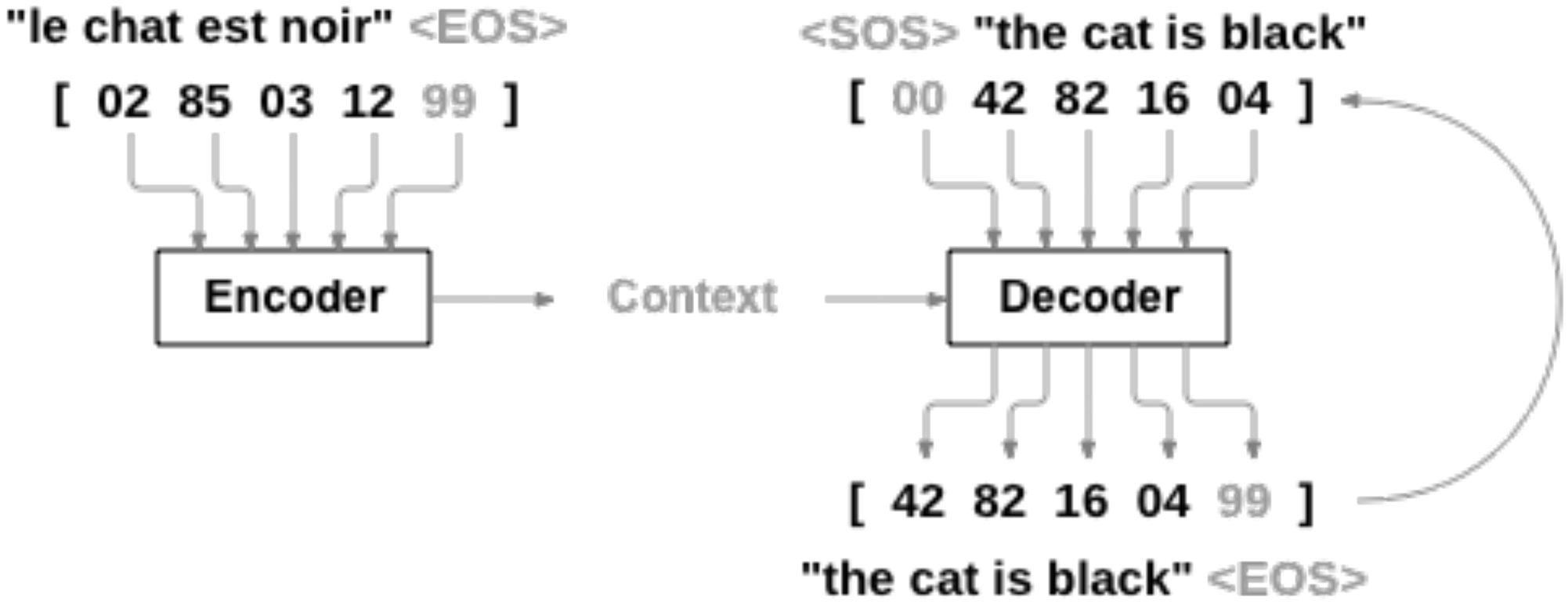

4x4

Sequence Modeling

Image credit: NASA/Goddard



Sequence Modeling: Recurrent Neural Networks



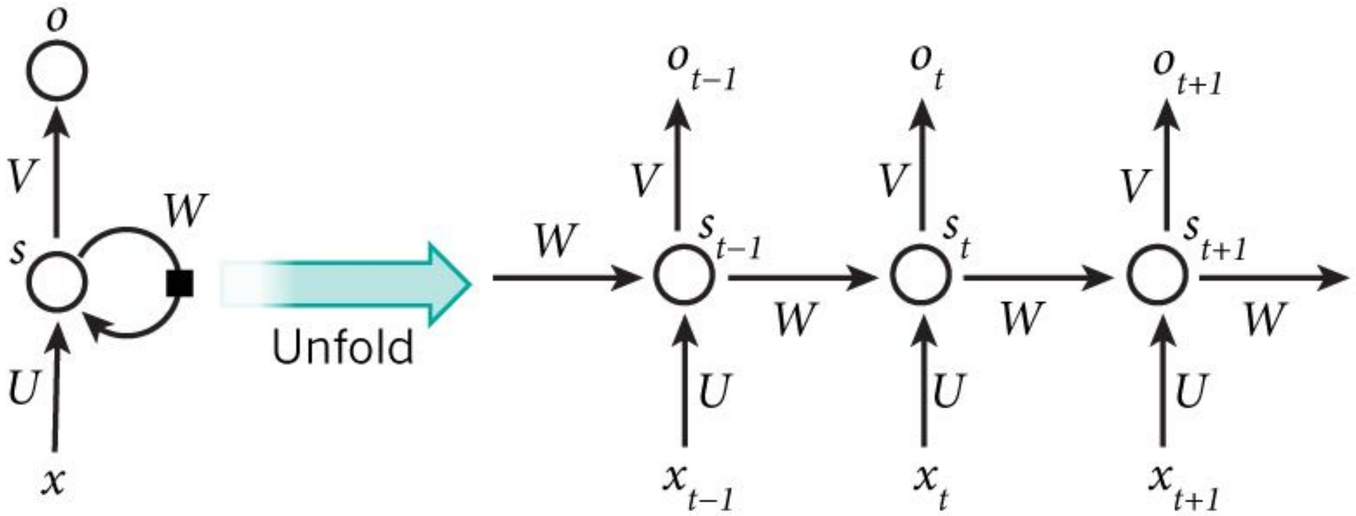
CNNs: spatial relevance
RNNs: temporal relevance

Unfolded Computational Graph of RNNs

Sequence $\left(x^{(t)}, x^{(t-1)}, x^{(t-2)}, \ldots, x^{(2)}, x^{(1)}\right)$

Recursive unfold $s^{(t)} = f\left(s^{(t-1)}; \theta\right)$

Output Sequence $\left(o^{(t)}, o^{(t-1)}, o^{(t-2)}, \ldots, o^{(2)}, o^{(1)}\right)$

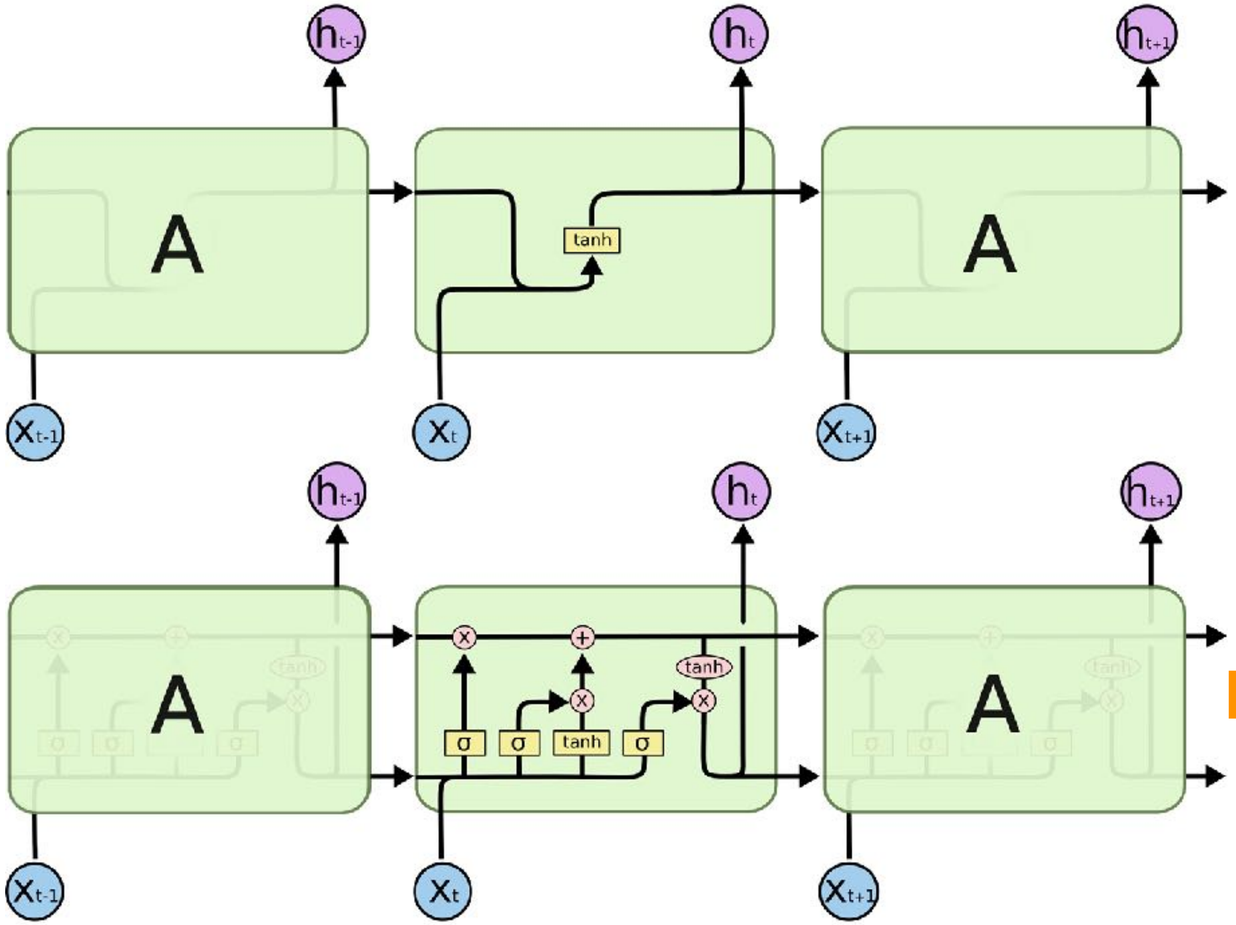


CNNs: Backprop through layers
RNNs: Backprop through time

The Problem of Long-term Dependencies

The clouds are in the *sky*.

I grew up in France… I speak fluent *French*.



Standard RNN

Long short-term memory (LSTM)

… ML/DL are built upon rigorous math theories
… ML/DL requires high-performance computing
… But we could just stand upon the shoulders of the giants


Caffe2


Microsoft
CNTK


TensorFlow


theano


Chainer


PYTORCH


dmlc
mxnet


K

Our systems (1/2)

Cartesius (Bull supercomputer):

- 40.960 Ivy Bridge / Haswell cores: 1327 TFLOPS
- 56GBit/s Infiniband
- 64 nodes with 2 GPUs each: 210 TFLOPS
  - NVIDIA Tesla K40m GPU
  - 12GB GDDR5
  - GPU-Direct RDMA
- Accelerator island: #4 Green500 (June 2014)
- Broadwell & KNL extension (Nov 2016)
  - 177 BDW and 18 KNL nodes: 284TFLOPS
- 7.7 PB Lustre parallel file-system

THE GREEN 500

Slide from Damian Podareanu's presentation

Our systems (2/2)

LISA (Dell cluster):

- 7856 cores (16 cores per node, Xeon E5-2650)
- Peak performance: 149 TFLOPS
- 24 * 4 NVIDIA 1080TI

HPC cloud:

- Virtual machines
- Up to 64 cores and 2TB RAM

The archive:

- Tape-storage for long-term storage
- Virtually unlimited space

Others:

- Grid
- Hadoop

Slide from Damian Podareanu's presentation